University of Alberta

Large-Scale Real-Time Electromagnetic Transient Simulation of Power Systems Using Hardware Emulation on FPGAs

by

Yuan Chen

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Power Engineering and Power Electronics

Department of Electrical and Computer Engineering

Spring 2012
Edmonton, Alberta

To my parents, my wife, and my sons,

for their love, endless support,

and encouragement.

Abstract

Real-time electromagnetic transient (EMT) simulation plays an important role in the planning, design, and operation of modern power transmission systems with adequate security and reliability due to increased load growth, interconnectivity, and stressful operating conditions. Real-time EMT simulators are widely employed for such applications as testing of advanced protective schemes for lines and generators, testing closed loop control systems either for conventional power systems or for power electronic based applications such as HVDC and FACTS, and for the training of system operators under realistic scenarios.

Real-time EMT simulation of large transmission networks requires very high computational capability. To meet the stringent real-time step-size constraints, a compromise is usually made between the size of the system simulated and the complexity of the component models. Taking advantage of the inherent parallel architecture, high density, and high clock speed, field programmable gate array (FPGA) has gained increasing popularity in high performance computation for various computationally expensive applications.

This thesis describes how FPGAs can be used for realizing real-time electromagnetic transient simulation of large-scale power systems using digital hardware emulation. Detailed parallel hardware modules for various power system components are described, including linear lumped RLCG elements, supply sources, circuit breakers. Hardware modules for transmission lines include traveling wave (Bergeron) model, frequency-dependent line model (FDLM), and universal line model (ULM). Various rotating electric machines are modeled using universal machine (UM) model. The network solution exploits sparse matrix techniques for improved efficiency. A novel parallelled EMT solution algorithm is described that accommodates the parallel FPGA architecture. For inclusion of nonlinear elements in power system, a parallel iterative nonlinear network solver is described that uses Newton-Raphson method both continuous and piecewise. Multiple FPGAs are utilized for real-time EMT emulation of large-scale power systems. A novel functional decomposition method is introduced to allocate the model components to the available hardware emulation modules in the FPGAs. All hardware arithmetic units designed are

deeply pipelined to achieve highest computation throughput. 32-bit floating-point number representation is used for high accuracy throughout the EMT simulation. The whole design is based on VHDL for portability and extensibility.

Various power system case studies are used to validate the proposed FPGA-based real-time EMT simulator. The captured real-time oscilloscope results demonstrate excellent accuracy and small simulation time-step of the simulator in comparison to the off-line simulation of the original systems in the ATP or EMTP-RV® off-line EMT programs.

Acknowledgements

I would like to express my sincere thanks to my supervisor *Dr. Venkata Dinavahi* for his full support, encouragement, and guidance for my research throughout the years I spent at the University of Alberta. His insightful guidance, passion and enthusiasm for the research has been an invaluable motivation for my life.

It is an honor for me to extend my gratitude to my Ph.D. committee members *Dr. José R. Marti (ECE, UBC)*, *Dr. Behrooz Nowrouzian*, *Dr. John Salmon*, *Dr. Andy Knight*, and *Dr. Dan Sameoto (MecE, U of A)* for reviewing my thesis and providing invaluable comments. Special thanks go to my colleagues and friends at the RTX-Lab: *Vahid Jalili-Marandi*, *Md Omar Faruque*, *Aung Myaing*, *Zhiyin Zhou*, *Jiadai Liu*, and *Yifan Wang*.

I wish to extend my deepest appreciation to my wife, Zhaohui, my sons, Leo and Nichola. Without their understanding, support, encouragement, and happiness, this thesis could not be finished.

Finally, financial help from NSERC, the University of Alberta, and Government of the Province of Alberta for my living in Edmonton during these years is greatly appreciated.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ALM	Adaptive Logic Module
ASIC	Application-Specific Integrated Circuit
ATP	Alternative Transients Program
CNR	Continuous Newton-Raphson
CORDIC	COordinate Rotation DIgital Computer
CPU	Central Processing Unit
DAC	Digital to Analog Converter
DSP	Digital Signal Processing (Processor)
EMT	ElectroMagnetic Transients
EMTP	ElectroMagnetic Transients Program
FDLM	Frequency-Dependent Line Model
FDNE	Frequency-Dependent Network Equivalent
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GJE	Gauss-Jordan Elimination
HIL	Hardware-in-the-Loop
HPC	High Performance Computation
IOE	Input/Output Element
IP	Intellectual Property
JTAG	Joint Test Action Group
LAB	Logic Array Block
LE	Logic Element
LUT	Look-up Table
NR	Newton-Raphson
PH	Processing Hardware

PLL	Phase Locked Loop
PNR	Piecewise Newton-Raphson
(S)RAM	(Static) Random Access Memory
RISC	Reduced Instruction Set Computer
TNA	Transient Network Analyzer
ULM	Universal Line Model
UM	Universal Machine
VF	Vector Fitting
VHDL	Very High-Speed Integrated Circuit Hardware Description Language
VLSI	Very Large-Scale Integrated Circuit

1 Introduction

Electromagnetic transients (EMT) are the temporary overvoltages and overcurrents caused by the change of power system configuration due to switching operation, fault, lightning strike, and other disturbances [1]. Typical EMT phenomena include lightning strikes on transmission lines, energization of transmission lines, shunt capacitor switching, interruption of small inductive currents, energization of transformer-terminated lines, motor starting, inrush current in transformer, linear resonance at fundamental or at a harmonic frequency, series capacitor switching and sub-synchronous resonance, load rejection, transient recovery voltage across circuit breakers, and very fast transients in gas-insulated bus ducts caused by disconnected operations [2]. It is called an electromagnetic transient because it involves predominantly interactions between the magnetic fields of inductance and the electric fields of capacitances in the system. It is distinguished from electromechanical transient which involves interactions between the mechanical energy stored in the rotating machines and the electrical energy stored in the network [3]. Electromagnetic transients can span a wide frequency range, from dc to several MHz; thus the EMT study is the most detailed study, and is different from other power system studies such as short-circuit, power flow, and transient stability. For example, phaser equations are replaced by differential equations; single-phase representation is replaced by multi-phase representation.

1.1 Electromagnetic Transient Simulation of Power Systems

Transients can damage component insulation, activate control or protective systems, and cause system interruption, thus EMT simulation plays an important role in the planning, design, and operation of modern power systems due to their increasing complexity. For ex-

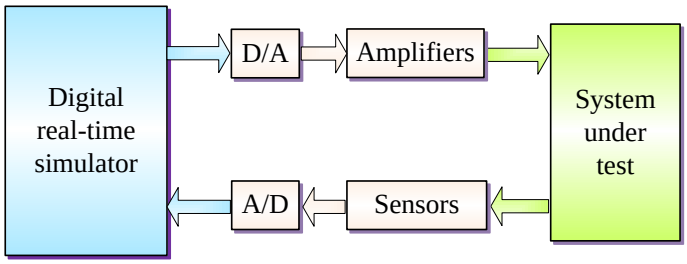


Figure 1.1: Hardware-in-the-loop configuration.

ample, it is used for determining component rating, insulation level, and design of protection schemes. Simulation can be accomplished in either off-line or real-time mode. For off-line transient simulation, the Electromagnetic Transients Program (EMTP) [4, 5] is widely accepted as an industrial standard. In the EMTP, the continuous models of all lumped elements in power systems are first discretized using the Trapezoidal rule of integration, and the transmission lines are represented using traveling wave model. Then the nodal analysis method is employed to solve for the node voltages of the network. Compared to other transient solution methods such as state variable analysis, the EMTP features simplicity (the whole system is reduced to a collection of equivalent resistance networks) and robustness (Trapezoidal rule of integration is numerically stable and robust). Over the years, more efficient models and algorithms have been developed, which enable the simulation of EMT in greater detail and efficiency. For instance, the modeling of transmission lines have evolved from the simple π representation, traveling wave model, frequency-dependant line model, to the most complicated universal line model. The widely used off-line EMT simulation software packages (referred as EMTP-type softwares) that offer a wide variety of modeling capability are ATP, PSCAD/EMTDC®, EMTP-RV®, MICRO-TRAN, NETOMAC®, etc.

Real-time simulation is desired for the testing of manufactured control and protection equipment in a hardware-in-the-loop (HIL) configuration. In the HIL simulation, as shown in Fig. 1.1, the simulation results have to be synchronized with the real-time clock to interface with the physical system under test such as a protective relay and machine drive. This is the only way that the physical device can be tested under realistic transient conditions of interest without the risk of putting it into the real world system. HIL simulation can be performed in two ways [6]: control hardware-in-the-loop (CHIL) and power hardware-in-the-loop (PHIL). In a CHIL simulation, the hardware under test is a controller, which exchanges signals with the simulated system at a low power level, while in a PHIL simulation the hardware under test involves actual power devices that require significant power flow between the hardware and the simulation system.

Real-time simulators were first developed on analog devices back in 1930s, known as transient network analyzers (TNAs). The analog TNA used scaled-down models of power

system components. For example, the frequency response of a long transmission line was approximated by cascading several analog π sections representing shorter line segments. The merits of analog TNA is that the simulation is naturally in real time because of its real component representation, while the disadvantages include high cost, large space, and long set up and changeover times. Over the last two decades, owing to the fast developing VLSI technology, the general purpose central processing unit (CPU) and digital signal processor (DSP) have been widely used to build the EMT simulators, known as real-time digital simulators. This type of simulators is fully digital featuring great flexibility, reduced cost, increased efficiency, thus enabling them to supplant analog TNAs. In the real-time digital simulator, the simulation time-step is the critical factor in order to reproduce the high frequency transients, for example, to investigate 10KHz voltage disturbance a 50μs time-step is required at least according to Nyquist criterion. Moreover, there is an ever increasing pressure to accommodate large system sizes. As a result, the more powerful computational hardware are always demanded and great efforts have been put into designing high performance digital real-time simulator, as discussed in the following survey.

1.2 Survey of Digital Real-Time EMT Simulators

Back in late 1980s, researchers were able to realize a real-time digital simulator using available digital hardware. [7, 8] presented a real-time digital simulator for the simulation of transmission lines using the Bergeron's traveling wave model and Marti's frequency-dependent line model, respectively. This simulator was based on a single DSP and interfaced into a TNA replacing the bulky and expensive part of the transmission line replica. [9] proposed a digital simulator for real-time protection relay testing. This simulator used IBM RISC 6000 processor for the real-time computation of network transients and a DSP for real-time instrument transformer transients. In [10–13], a dual DSP architecture was proposed for real-time transmission line simulation, where each DSP simulated one end of the transmission line (sending-end or receiving-end), as shown in Fig. 1.2. In contrast to customized hardware architecture, [14] realized real-time simulation based on the standard workstation equipment. It achieved the simulation time-step ranging from 38μs to 107μs for systems from 18 to 30 nodes using an IBM RISC 6000 workstation. In the late 1990s to mid 2000s, the PC-cluster architecture has drawn increasing attractions [15–17]. It uses a cluster of standard PCs interconnected by high-speed interface cards. This architecture features cost effectiveness, great flexibility, and easy extensibility.

During the same time, constant research interest and effort have led to the development of commercial digital real-time simulators. Some commercial available digital real-time simulators are as follows:

- The RTDS®, from RDTS Technologies Inc., was the first commercial real-time digital simulator proposed in 1991 for testing relays in real time [18, 19]. The latest

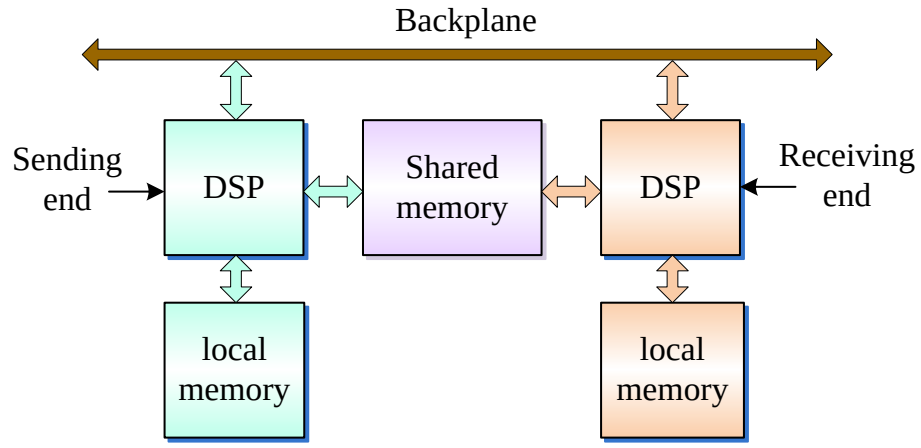


Figure 1.2: A dual-DSP architecture for transmission line simulation [13].

RTDS simulator adopts a custom parallel processing architecture. Each hardware unit (called a Rack) typically consists of several Giga Processor Cards (GPC) which are powered by IBM PowerPC 750GX RISC processors, and one Giga Transceiver Workstation InterFace Card (GTWIF) which handles the communication between the rack and the host computer [20].

- HYPERSIM® [21, 22], from Hydro-Quebec, is a full digital real-time simulator based on large parallel supercomputers such as the SGI Origin 3800 scaled from 2 to 512 processors. Now it is adapted to a new PC-cluster platform [23].

- eMEGAsim® [24, 25], from OPAL-RT Technologies Inc., utilizes the commercial-off-the-shelf (COTS) multi-core processor (Intel or AMD) module along with fast on-chip inter-processor shared-memory communication. The simulated network is built on the host PC using MATLAB/SIMULINK in block diagram format, or custom defined C S-function. The compiled code is then downloaded into target PCs for simulation.

- NETOMAC® [26, 27], from SIEMENS AG, is a PC-based real-time simulator originally for relay testing. It runs the NETOMAC program to generate the transient output to a interface card connected to the tested relay.

To summarize, there are three points worthy of note:

1. The evolution of digital real-time simulator is from the fully custom DSP-based, to the commercial supercomputer-based, to the low-cost standard PC-based. Although the current DSP and general purpose CPU are very powerful, essentially they are sequential hardware devices. Massive parallel processing using such hardware is still a bottleneck.

2. In order to simulate large network in real time, parallel processing is always necessary. This is achieved by using parallel processing hardware to share the burden of

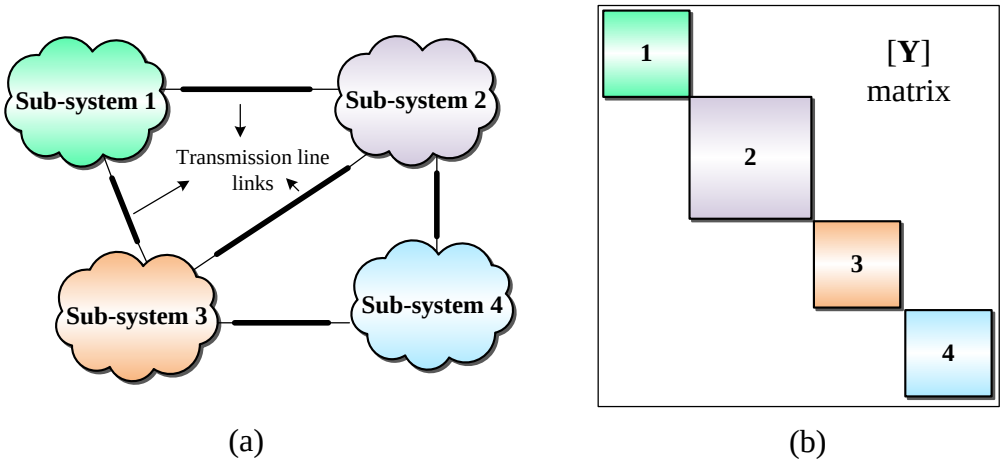


Figure 1.3: (a) Partitioning of a large power network into sub-systems, and (b) its block diagonal format system admittance matrix [14].

computation. To do this, the simulated system is first partitioned into smaller sub-systems illustrated in Fig. 1.3 (a), and all sub-systems are linked by transmission lines. Due to the traveling wave delay of the linking transmission line, each sub-system is decoupled with the others, thus the system admittance matrix is in a block diagonal format, as shown in Fig. 1.3 (b). Then each processing hardware is made responsible for calculation of the quantities within a sub-system. Before proceeding to the next time-step the neighboring hardware need to exchange data of the both sides of the linking transmission line.

3. Another important approach[1] to accommodate large network sizes while maintaining sufficient accuracy for real-time simulation [28–30] is to divide the system into a *Study Zone*, a restricted part of the system where the transient phenomena occur and whose components must be fully characterized including any nonlinear and time-variant elements, and an *External System* which encompasses the rest of the system as shown in Fig. 1.4 [30]. The external system is represented by a linear equivalent network, i.e., a frequency-dependent network equivalent (FDNE). However, deriving an accurate FDNE model is a finely-tuned art requiring skill and significant setup time. In contrast to a full-scale representation, there is also a certain degree of loss of one-to-one mapping between the original system physical layout and the simulator architecture.

[1]This material has been published: X. Nie, Y. Chen, and V. Dinavahi, "Real-time transient simulation based on a robust two-layer network equivalent", *IEEE Trans. on Power Systems*, vol. 22, no. 4, pp. 1771-1781, November 2007.

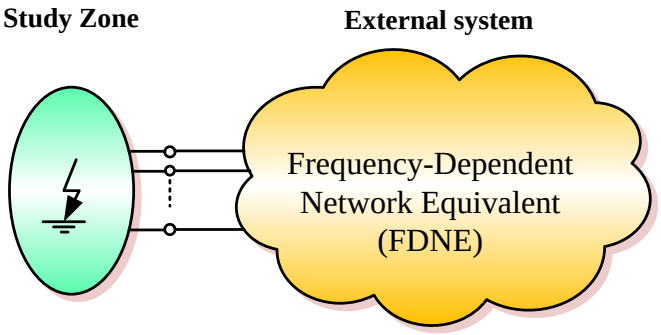


Figure 1.4: Study zone and external system in FDNE [30].

1.3 Motivation for this Work

The endless growing complexity and size of modern power systems demand faster and higher performing real-time digital simulators. The ultimate performance of any real-time simulator depends to a large extent on the capabilities of the underlying hardware. Field Programable Logic Array (FPGA) offers viable alternative for speeding up the digital simulator without sacrificing accuracy. Unlike general purpose CPU or DSP which is basically a sequential device, FPGA is a digital hardware device which allows true parallel processing, supporting multiple simultaneous threads of execution. Moreover, the FPGA is a fully user configurable device, i.e., it can be configured to fit any specific application in order to maximize performance. It is this inherent hardware parallelism and configurability that makes the FPGA suitable for the EMTP hardware implementation.

FPGAs are increasingly being used to design high-end computationally intense applications. In the power engineering field, they have been employed in digital simulation of electrical machines [31–35], power electronics [36–39], digital control [40–46], pulse-width modulation [47–50], and protective relays [51, 52]. However, in the general simulation of electromagnetic transient, FPGAs have not been fully used to replace the DSP and general purpose CPU. A complete real-time EMT simulator based on FPGA is still challenging. The first challenge is that the FPGA design is quite difficult compared to software programming using the high-level languages like C/C++. In the FPGA design, every calculation needs to be assigned a specific hardware module. Many independent hardware modules may work together in parallel only if the algorithm itself can be performed in parallel. To implement a complicated algorithm such as the universal line model great effort is needed to analyze the algorithm, find the internal parallelism, design respective hardware modules, and interconnect hardware modules. It would be much more complicated if multiple FPGAs are employed in order to simulate large-scale power systems due to the massive and high-speed inter-FPGA data sharing. The second challenge is that the user friendly interface (GUI) for the FPGA-based transient simulator is still not available. Running a real-time simulation on the FPGA-based simulator is restricted to experienced

personnel. This thesis focuses on the first challenge to design a fully FPGA-based real-time digital simulator for electromagnetic transient simulation of large-scale power system using detailed component models. In the knowledge of the author, it is the first time that the EMT transient simulator is realized in FPGA using most detailed models for most of power system components.

1.4 Research Objectives

To realize real-time electromagnetic transient simulation of large-scale power systems using FPGAs, the main research objective of this thesis are listed as follows:

1. Individual components of power systems have to be modeled and implemented. The complexity of models determines the accuracy of simulation results. However, implementing more complicated models implies more logic resource utilization of FPGA and longer execution times. Based on the available FPGA resource, required simulation time-step, and the size of simulated network, the models of components which can be implemented need to be carefully chosen. The implemented power system components include transmission lines modeled using frequency-dependent line model (FDLM) and universal line model (ULM), linear lumped RLCG elements, supply sources, circuit breakers, rotating machines modeled using universal machine (UM) model, nonlinear inductances and surge arresters.

2. The linear network needs to be solved efficiently using sparse matrix technique.

3. Solving nonlinear network accurately in real time is hard to achieve in the CPU and DSP based digital simulator. An real-time iterative Newton-Raphson nonlinear solver on the FPGA is required.

4. The traditional EMTP algorithm is essentially a sequential procedure. In order to take advantages of parallel architecture of FPGA, the EMTP algorithm needs to be reformulated. The possible parallel computational tasks have to be extracted and allocated into parallel computational hardware.

5. To simulate large-scale power systems, a single FPGA is no longer able to meet real-time requirements; multiple FPGAs are necessary. In the multi-FPGA architecture, fast data transfer between FPGAs using limited FPGA input/output pins becomes critical. How to allocate the hardware modules for various components of the power system in such a multi-FPGA architecture is very important.

6. The parameters of simulated network are extracted from the netlist generated by off-line EMTP software ATP or EMTP-RV . Then they are sorted according to different types of components and processed for the requirement of the corresponding hardware. Finally they are translated to 32-bit floating-point format and saved as various

RAM initial data files. When the FPGAs are powered up, the parameters are ready in RAMs. A Matlab script file is developed to fulfil these tasks.

7. The implemented hardware modules are made of many basic arithmetic units such as adders/subtractors, multipliers, RAMs, and registers. To achieve high performance these individual hardware units need to be combined properly to form a pipeline. Parallelled and pipelined computation schemes are key for real-time EMT simulation on FPGA.

8. The proposed real-time EMT simulation hardware designs need to be validated by various transient case studies. The real-time results captured from oscilloscope are compared with off-line simulation results from ATP or EMTP-RV.

1.5 Thesis Outline

This thesis consists of seven chapters. Other chapters are outlined as follows:

Chapter 2 gives a general introduction to FPGA technology including its architecture and design tools and design flow. Some FPGA design issues are also discussed.

Chapter 3 describes a real-time EMT simulator based on a single FPGA. Details in modeling basic power system components including frequency-dependent transmission lines (FDLM), linear lumped RLCG elements, supply sources, circuit breakers, and their FPGA implementations are provided. The EMTP algorithm is parallelized to fit into the parallel architecture of the FPGA.

Chapter 4 describes a real-time hardware emulation of nonlinear elements in power system using an iterative Newton-Raphson method along with the compensation method. Sparse matrix techniques, parallel Gauss-Jordan elimination are exploited for improved efficiency.

Chapter 5 extends the transmission line model from FDLM to the most comprehensive and accurate universal line model (ULM). This chapter also presents the hardware emulation of the universal machine (UM) model.

Chapter 6 presents the real-time hardware emulation of large-scale power systems using a multi-FPGA architecture. A novel functional decomposition method is introduced to allocate the model components to the hardware emulation modules. A three-phase 420-bus power system is simulated in a 10-FPGA hardware platform.

Chapter 7 describes the conclusions and future work.

2
FPGA Background

The implementation of digital systems can be based on microprocessors, reconfigurable hardware, or application specific integrated circuits (ASICs). On the one hand, microprocessor implementation is fully software programmable thus very flexible but with low hardware efficiency; on the other hand, ASIC gives highest performance at the cost of loss of flexibility, high development cost, and long time to market. The reconfigurable hardware fills the gap effectively between microprocessors and ASICs. The configurable hardware is an off-the-shelf device featuring custom reprogrammable, low development cost, and short development cycle. Among reconfiguration hardware FPGA is the most widely used.

In this chapter, an introduction to the FPGA technology is presented. First the simplified FPGA architecture is described briefly. Then the FPGA design tools and design flow are discussed. Finally some important issues in FPGA hardware design are also discussed.

2.1 FPGA Architecture

In general, the FPGA is a two dimensional array of programmable logic building blocks interconnected by a matrix of wires and programmable switches. Each logic building block performs a basic logic function. The programmable switches control configuration of logic building blocks and the interconnection of the wires, thus achieving field programmability. The switches are physical transistors controlled through different programming technologies which classify FPGA into three categories: SRAM-based, fuse-based, and EPROM/EEPROM/flash-based, with the SRAM-based FPGA being the most popular and of interest for most applications. In the SRAM-based FPGA the configuration file is stored in the distributed SRAM. The configuration is usually fast, but reconfiguration is

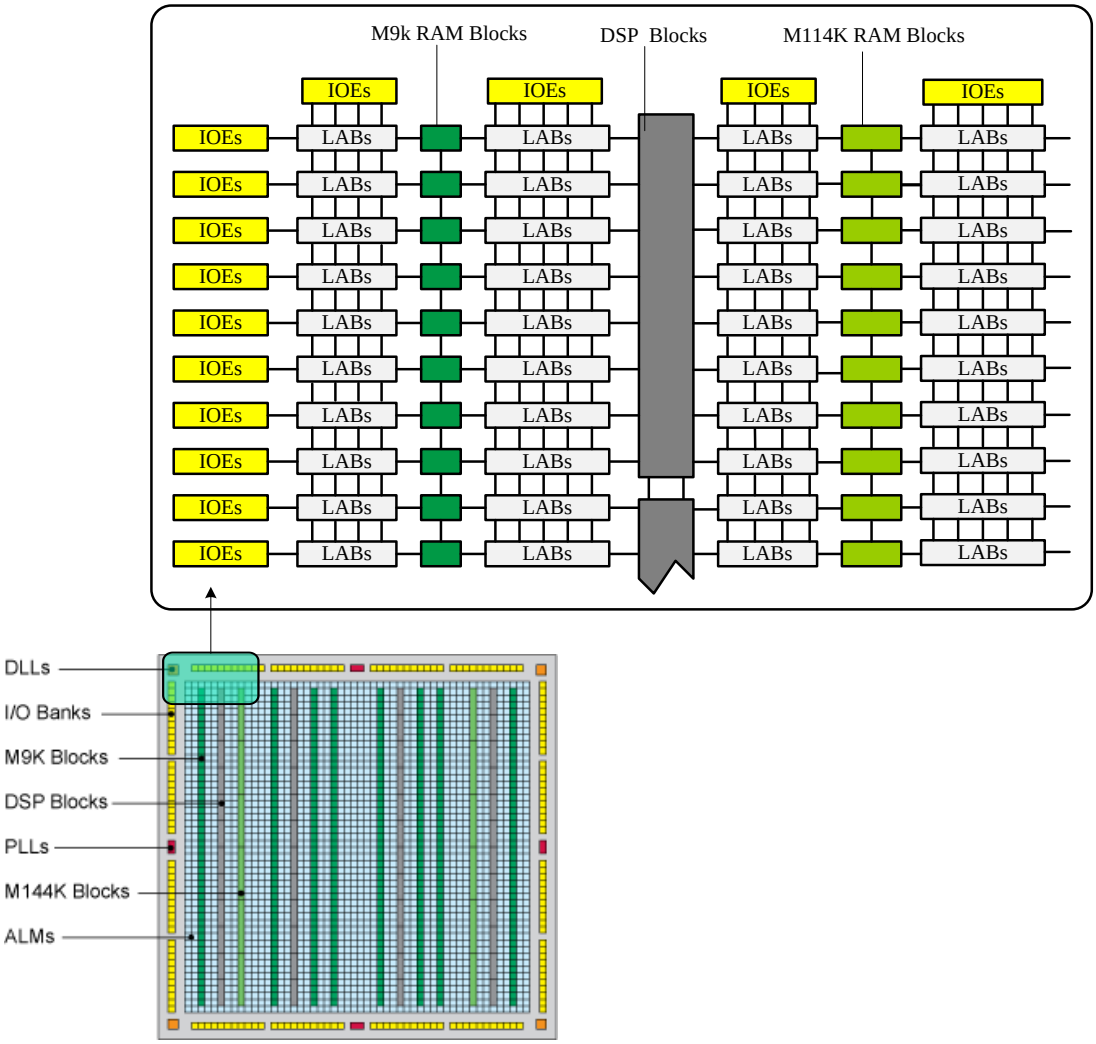


Figure 2.1: Altera Stratix III FPGA architecture block diagram [53].

required when the power is turned off due to the volatility of the SRAM. Modern FPGA also contains many blocks of memory, and other specialized functional circuits such as digital signal processing (DSP) blocks, phase locked loop (PLL), and even soft processor cores. The major SRAM-based FPGA vendors are Altera® and Xilinx® which share over 60% of the market. Since Altera Stratix™ III FPGA is utilized in the design of the real-time EMT simulator, its architecture is described in more detail as follows.

As shown in Fig. 2.1, the Altera Stratix III device contains a two-dimensional row- and column-based architecture including a large amount of logic array blocks (LABs), memory blocks, and DSP blocks which are interconnected by a series of column and row interconnects [53]. Table 2.1 lists the main logic resource of Stratix III EP3SL340 FPGA.

Table 2.1: Main logic resource of Altera Stratix III EP3SL340

Feature	EP3SL340
Equivalent logic elements (LEs)	340,000
Adaptive logic modules (ALMs)	135,000
M144k RAM blocks	48
M9k RAM blocks	1,040
MLAB blocks	6,750
Total RAM Kbits	16,272
DSP blocks (18x18-bit multipliers)	72 (576)
Phase locked loops (PLLs)	12
Maximum user I/O pins	1,760

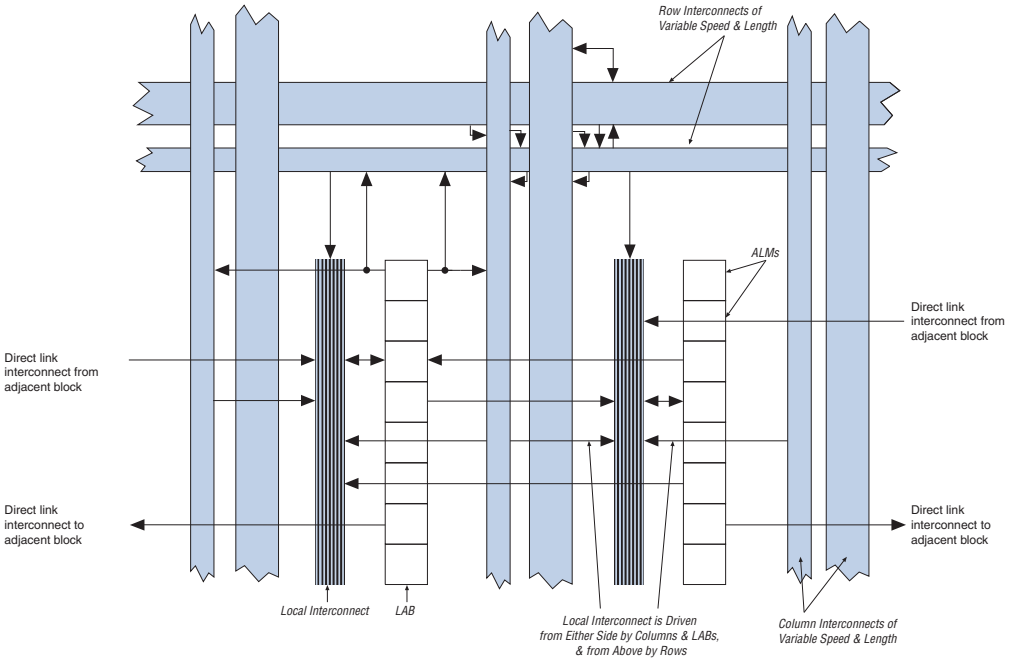


Figure 2.2: LAB structure [53].

2.1.1 Logic Array Blocks (LABs) and Adaptive Logic Modules (ALMs)

Each LAB consists of 10 ALMs, carry chains, shared arithmetic chains, LAB control signals, local interconnect, and register chain connection lines as shown in Fig. 2.2. The local interconnect transfers signals between ALMs in the same LAB. Look-up table (LUT) chain connections transfer the output of one ALM's LUT to the adjacent ALM for fast sequential LUT connections within the same LAB. Register chain connections transfer the output of one ALM's register to the adjacent ALM's register within an LAB [53].

ALM is the core and basic logic building block in the Altera Stratix III architecture. As shown in Fig. 2.3, each ALM contains a combinational logic, two registers, and two adders to implement arithmetic logic. The combinational logic has 8 inputs and contains

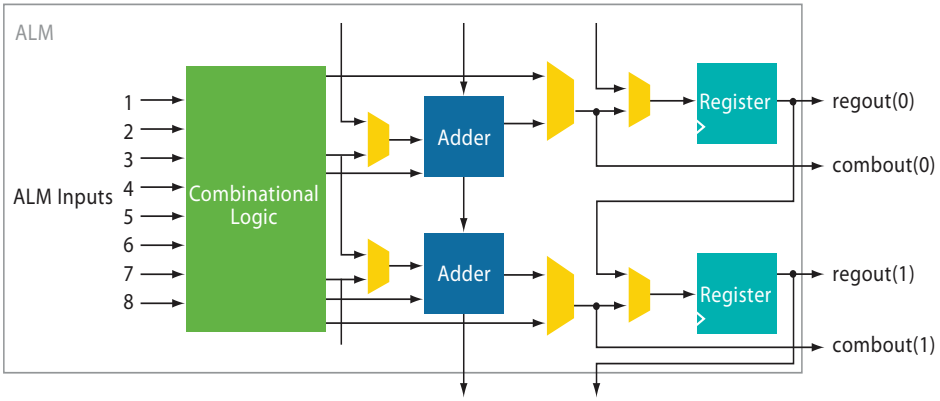


Figure 2.3: ALM block diagram [53].

two adaptive four-input look-up-tables (ALUTs) which support a variety of combinational configurations such as fully 6-input LUT, or two independent 4-input functions. Historically Altera used logic elements (LEs) as its basic logic building blocks in the FPGA. A LE contains a 4-input LUT, a register, and special-purpose carry circuitry for arithmetic circuits. Thus an ALM is able to implement 2.5 LEs equivalently.

2.1.2 Memory Blocks

Stratix III memory contains three types of memory blocks, namely M144k (144-Kbit) RAM blocks, M9K (9-Kbit) RAM blocks, and MLAB (320-bit) blocks. As can be seen in Fig. 2.1 and Table 2.1, these memory blocks are placed in different locations in the FPGA with various quantities. They can provide dedicated true dual-port, simple dual-port, and single-port RAM, ROM, and FIFO buffers with fully custom word width and length configuration. Fig. 2.4 shows a single-port RAM (a) and a true dual-port RAM. These massive, high speed access (up to 600MHz), and parallelized memory blocks are extremely useful in parallel computations as discussed in detail later.

2.1.3 Digital Signal Processing Blocks (DSPs)

High-speed DSP blocks provide dedicated implementation of multipliers (faster than 300 MHz), multiply-accumulate functions, and finite impulse response (FIR) filters. It can implement up to either eight full-precision 9x9-bit multipliers, four full-precision 18x18-bit multipliers, or one full-precision 36x36-bit multiplier with add or subtract features. DSP blocks are grouped into two columns in each device. Fig. 2.5 shows the simplified DSP block structure.

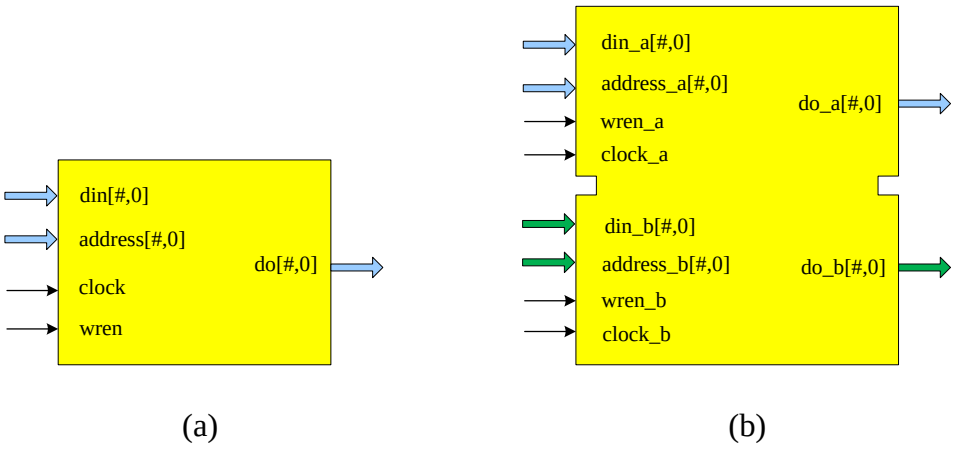


Figure 2.4: (a) Single-port RAM, and (b) true dual-port RAM.

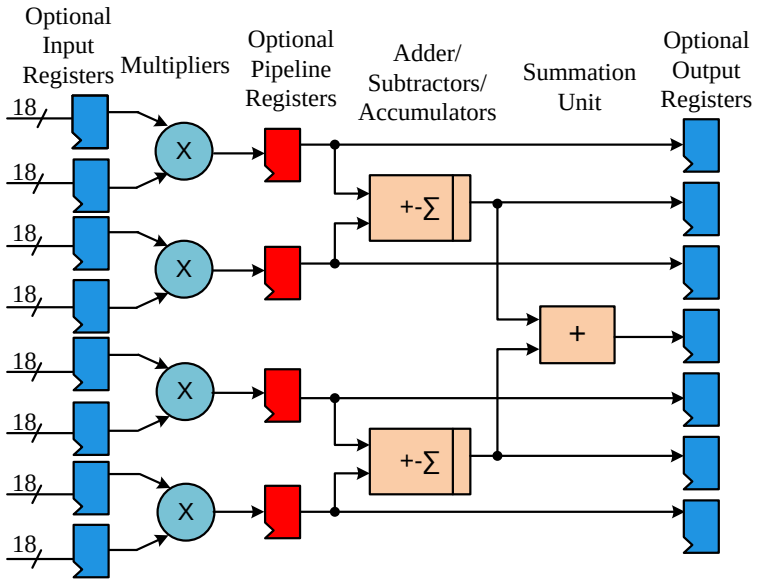


Figure 2.5: DSP block structure [53].

2.1.4 Phase-Locked Loops (PLLs)

Stratix III devices offer up to 12 PLLs that provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces. The main goal of a PLL is to synchronize the phase and frequency of an internal or external clock to an input reference clock. Fig. 2.6 shows the Stratix III PLL block diagram.

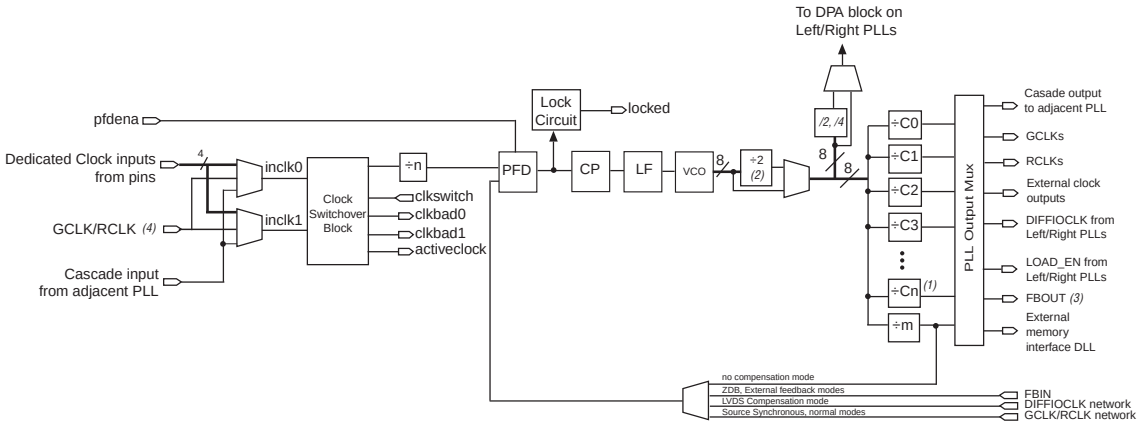


Figure 2.6: Stratix III PLL Block Diagram [53].

2.1.5 Input/Output Elements (IOEs)

IOEs are special logic blocks at the periphery of device for external connections. Each Stratix III device I/O pin is fed by an I/O element (IOE) located at the end of LAB rows and columns around the periphery of the device. I/O pins support numerous single-ended and differential I/O standards. Each IOE contains a bidirectional I/O buffer and six registers for registering input, output, and output-enable signals. Fig. 2.7 shows the IOE structure.

2.2 FPGA Design Tools and Design Flow

Along with the evolution of FPGA density and speed the FPGA design tools have also achieved a mature level being able to support all aspects of the entire FPGA design. Meanwhile, abundant vendor and third-party intellectual property (IP) cores make FPGA design fully self-contained. The Altera Quartus II software, for example, provides a full-integrated development environment for design, verification, debugging, and implementation. The generic FPGA design flow mainly consists of design entry, synthesis, implementation, verification, and configuration, as shown in Fig. 2.8.

Design Entry- The entry design can be based on schematic or hardware description language (HDL). The schematic-based FPGA design is used when the schematic of the design is available. Most commonly the entry design is done using HDL, especially when the design is algorithm oriented. The HDL is a powerful language to describe the behavior or structure of a digital system. It supports top-down or bottom-up design methodology. Systems can be described at different levels from architecture level to gate level. Two widely used HDLs are very high speed integrated circuit hardware description language (VHDL) and Verilog HDL. For commonly used hardware components such as memory block, IO interface, arithmetic unit, the venders' IP cores are the best choice since these

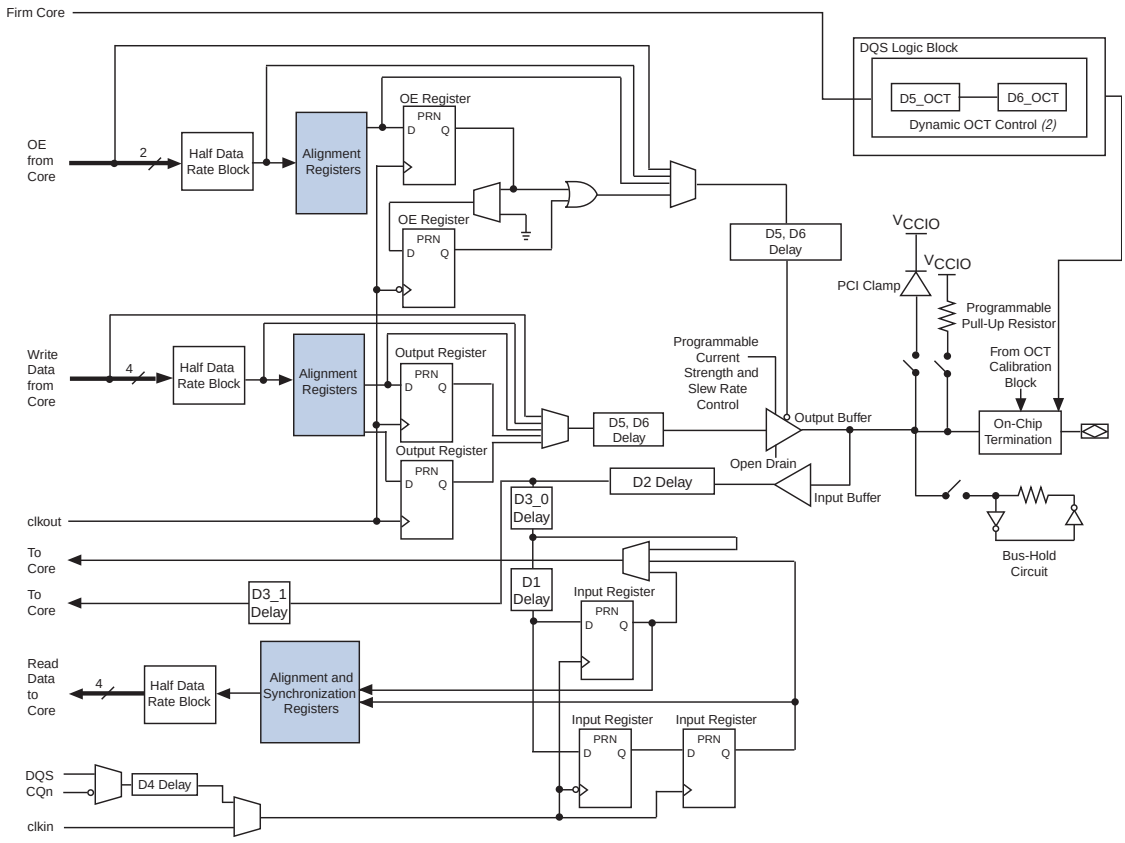


Figure 2.7: Structure of IOE [53].

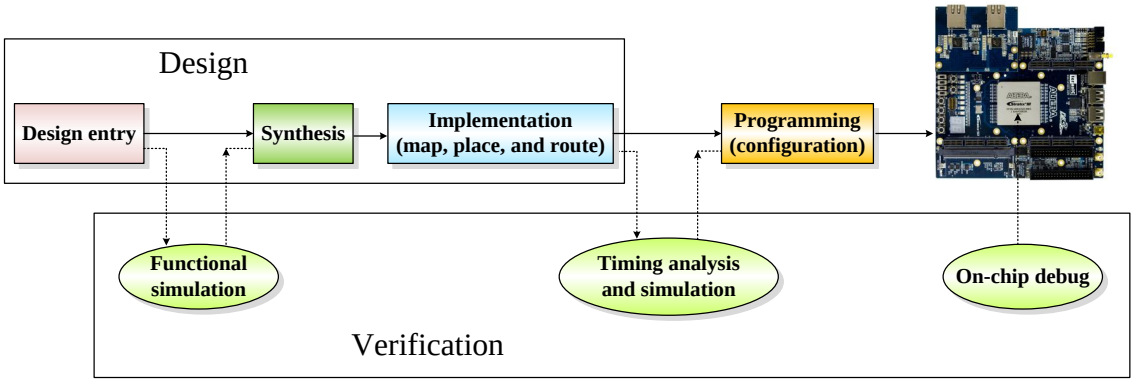


Figure 2.8: General FPGA design flow.

cores are well designed, fully tested, and optimized to the FPGA utilized. For example, MegaWizard® and SOPC builder® design tools are included in Alter Quartus II software.

Functional simulation- The functional simulation is employed to verify the entry design before the synthesis process. It is helpful to ensure the required functionality is sat-

isfied by checking the output simulation waveforms before starting the time consuming synthesis process.

Synthesis- This process translates entry design file (source HDL codes) into a device netlist format. The netlist describes the digital system in terms of basic logic components such as logic gates, registers, multiplexors, etc.

Implementation- This process is comprised of three sub-processes: *Map*, *Place*, and *Route*. The Map sub-process divides the whole design into the basic logic blocks such as ALM, IOE, DSP. The Place sub-process assigns them to physical location. The Route sub-process selects wire segment and switches for interconnection.

Timing analysis and simulation- Design verification at this stage can be done using timing analysis and timing simulation. It is a very important step to guarantee that the FPGA device's functionality meets all required timing constrains for the design.

Device Programming- After implementation, the design is converted to a format that can be accepted by the FPGA. This file which is called a bitstream is the configuration file of the FPGA. It is downloaded into the FPGA using the JTAG interface from a host computer. Now the FPGA design is finished and it is ready for use.

2.3 FPGA Design Issues

2.3.1 Data Representation

What data format to use is the first question that needs to be addressed for any FPGA implementation, since it will affect the accuracy of computation and logic resource utilization. Basically there are two types of number system: fixed-point number and floating-point number. Comparing with the fixed-point number, on the one hand, the floating-point number has advantages of dynamic range, high accuracy; so it is widely used in high precision arithmetic. On the other hand, floating-point computation takes longer execution time and requires more hardware resources to implement. For example, a float-point multiplier needs 2 adders, 1 multiplier, and usually has more than 5 clock cycle latency. That is why the implementation of floating-point arithmetic was difficult until recently, i.e., until the increased density and speed of modern FPGA.

In this thesis, the 32-bit (single-precision, IEEE Standard 754) floating-point format is utilized in the FPGA-based real-time EMT simulation. As shown in Fig. 2.9, a 32-bit floating-point number consists of 1 *sign* bit, 8 *exponent* bits, and 23 *fraction (mantissa)* bits. A floating-point number v is described as follows

$$v = (-1)^{sign} \times 2^{(exponent - exponent\,bias)} \times (1.mantissa), \tag{2.1}$$

where the exponent bias is 127. The range of v is approximately 10^{-38} to 10^{38}.

In the EMTP, the accumulation ($\sum$) is commonly used in the convolution integration and matrix multiplication . However, a floating-point accumulation typically consumes

Figure 2.9: 32-bit floating-pointer number format.

more FPGA resources and has a very long latency [54]. For this reason, a fixed-point accumulator is used instead of a floating-point accumulator. The fixed-point number format is 40.100 with 40 bits integer and 100 bits fraction to guarantee both the range and precision. All data in FPGA are saved and processed in floating-point number format. Only when accumulation is required, the floating-point number is converted to fixed-point format, the accumulation result is then converted back to floating-point format.

2.3.2 Parallelism

The most attractive feature of an FPGA is its intrinsic massive parallel architecture. It means that FPGAs can be partitioned and configured into a large number of parallel-processing units and all units can process their data simultaneously. A general purpose CPU or DSP, on the other hand, is a sequential device in which one or several instructions are executed sequentially. The process of an instruction includes fetch, decode, execution, and write-back in one or more clock cycles. As mentioned earlier, modern FPGAs contain many blocks of memory which can be configured to be of various types with any bus width and depth. Thus, many independent memory units can be accessed concurrently, whereas in a general purpose CPU or DSP, all memory and buses are shared in a time division multiplexing mode. For computationally complex applications, the dedicated high-performance DSP blocks provide fast and flexible computation engines. Many DSP blocks can be configured together to perform many addition, subtraction, multiplication, and summation with any data bandwidth simultaneously. Meanwhile, FPGAs also provide very wide IO bandwidth, for instance, Altera Stratix III EP3SL340 has 1,760 user I/O pins which make the parallel input/output possible. An example shown in Fig. 2.10 is used to explain the difference between FPGA and general purpose CPU or DSP for implementing a function. In the FPGA implementation scheme (Fig. 2.10 (a)), 2 hardware multipliers and 1 adder are implemented and connected together. In the CPU implementation (Fig. 2.10 (b)), a set of instructions are first developed, then fetched, decoded, and executed one after the other sequentially. It is clear that the FPGA implementation is space-oriented (each calculation is processed in its own hardware units in parallel), while the CPU implementation is time-oriented (each instruction is processed in a single hardware sequentially).

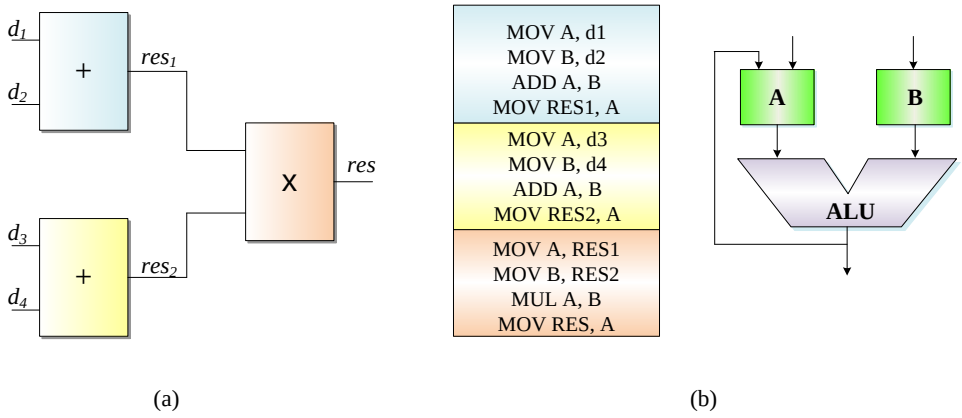


Figure 2.10: An example showing the different implementations in (a) FPGA, and (b) CPU/DSP.

2.3.3 Pipelining

In the paralleled architecture, the pipelining technique is another important strategy for improving hardware performance. In pipeline scheme, a function is divided into several stages depending on the number of operations, in which different operations are performed. Registers are then inserted between these stages to separate the operations. Thus, data can march through the operations at every clock. Although a pipeline has the cost of *latency* which is the input to output delay in terms of clock cycles, the key merit for a pipeline is its computational throughput, which is defined as the number of operations that can be performed per unit of time.

An example to illustrate the pipeline scheme is the convolution function which is widely used in digital signal processing. A convolution function defined by

$$C = \sum_{i=1}^{N} c_i = \sum_{i=1}^{N} a_i x_i + b_i y_i, \tag{2.2}$$

can be hardware implemented in FPGA shown in Fig. 2.11. The pipeline consists of 4 stages. In *Stage 1*, all input data are accessed from the RAM banks simultaneously. The multiplications are carried out at *Stage 2* and the addition is done at *Stage 3*. The final summation is done at *Stage 4*. As shown in the figure, the data flows into the pipeline every clock and is processed independently at each individual stage. In this case, the latency is 4 clock cycles while the throughput is 1 result per clock cycle. If the convolution operation is performed without the pipeline, the latency and throughput would be 4 clock cycles and 1 result per 4 clock cycles, respectively.

2.4 Summary

With the fast increasing density and speed, FPGAs have been gaining great interest for high performance computation (HPC) applications. The intrinsic massive parallel archi-

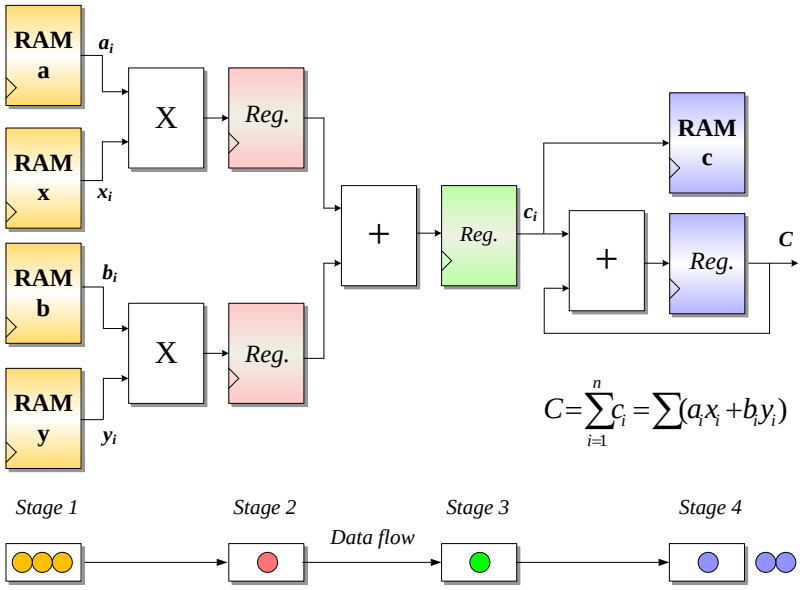


Figure 2.11: An example of convolution implemented in FPGA [57].

tecture, fully custom configuration, hardware pipelining support, abundant IP cores, and complete development environment features make the FPGA outperform the general purpose CPU or DSP for most digital signal processing applications. Therefore, the FPGA is chosen in this thesis for real-time EMT simulation which involves many computationally intensive algorithms. This chapter provided a brief overview of the hardware architecture, tools and procedures, and issues involved when using FPGAs for digital system design.

3

FPGA-Based Real-Time EMT Simulator

In this chapter[1] the process of implementing a real-time electromagnetic transient simulator on a single FPGA is described. The FPGA is used to emulate the basic power system components including transmission lines, linear lumped RLCG elements, supply sources, circuit breakers. The modeling of each component is described before giving the FPGA implementation details. The central component of this simulator is the frequency-dependent line model (FDLM) which is a widely used accurate line model. The network is solved using sparse matrix technique for improved efficiency. The EMTP algorithm is analyzed, and the inherent parallel computations are realized in the parallel hardware architecture of FPGA. All developed hardware computational units are fully pipelined to achieve the highest throughput. A case system consisting of 15 transmission lines, 4 generators, and 8 loads is simulated for two transients. The captured real-time simulation waveforms from the oscilloscope show the detailed agrement with the ATP off-line results.

3.1 Frequency-Dependant Line Model

3.1.1 FDLM Model Formulation

Taking into account the frequency dependence of parameters and the distributed nature of losses in the transmission lines, Marti's frequency-dependent line model (FDLM) [55,56] is widely used and simulated [8,12,13]. This model is based on the well known line equations

[1]Material from this chapter has been published: Y. Chen and V. Dinavahi, "FPGA-based real-time EMTP", *IEEE Trans. on Power Delivery*, vol. 24, no. 2, pp. 892-902, April 2009.

in the frequency-domain as follows

$$\begin{aligned} V_k(\omega) &= \cosh[\gamma(\omega)\ell]V_m(\omega) - Z_c(\omega)\sinh[\gamma(\omega)\ell]I_m(\omega), \\ I_k(\omega) &= \frac{1}{Z_c(\omega)}\sinh[\gamma(\omega)\ell]V_m(\omega) - \cosh[\gamma(\omega)\ell]I_m(\omega), \end{aligned} \tag{3.1}$$

where $V_k(\omega)$, $V_m(\omega)$, $I_k(\omega)$ and $I_m(\omega)$ are the frequency-domain quantities corresponding to sending-end ('k') and receiving-end ('m') voltages and currents, respectively; ℓ is the line length; $Z_c(\omega)$ and $\gamma(\omega)$ are frequency-dependent characteristic impedance and propagation function defined as

$$\begin{aligned} Z_c(\omega) &= \sqrt{\frac{R(\omega) + j\omega L(\omega)}{G + j\omega C}}, \\ \gamma(\omega) &= \sqrt{(R(\omega) + j\omega L(\omega))(G + j\omega C)}, \end{aligned} \tag{3.2}$$

where $R(\omega)$, $L(\omega)$, G, and C are series resistance, series inductance, shunt conductance, and shunt capacitance, respectively. New equations that relate currents and voltages in the frequency-domain are then introduced [56]:
forward traveling functions

$$\begin{aligned} F_k(\omega) &= V_k(\omega) + Z_c(\omega)I_k(\omega), \\ F_m(\omega) &= V_m(\omega) + Z_c(\omega)I_m(\omega), \end{aligned} \tag{3.3}$$

and backward traveling functions

$$\begin{aligned} B_k(\omega) &= V_k(\omega) - Z_c(\omega)I_k(\omega), \\ B_m(\omega) &= V_m(\omega) - Z_c(\omega)I_m(\omega). \end{aligned} \tag{3.4}$$

By eliminating $V_k(\omega)$, $V_m(\omega)$, $I_k(\omega)$ and $I_m(\omega)$ from (3.1) and (3.4), we obtain

$$\begin{aligned} B_k(\omega) &= A_1(\omega)F_m(\omega), \\ B_m(\omega) &= A_1(\omega)F_k(\omega), \end{aligned} \tag{3.5}$$

where

$$A_1(\omega) = e^{-\gamma(\omega)\ell} = \frac{1}{\cosh[\gamma(\omega)\ell] + \sinh[\gamma(\omega)\ell]}. \tag{3.6}$$

Equations (3.4) give the Thévenin equivalent network shown in Fig. 3.1. The voltage sources $b_k(t)$ and $b_m(t)$ are the time-domain forms of $B_k(\omega)$ and $B_m(\omega)$, which are convolution integrals of (3.5) as

$$\begin{aligned} b_k(t) &= \int_\tau^\infty f_m(t-u)a_1(u)du, \\ b_m(t) &= \int_\tau^\infty f_k(t-u)a_1(u)du, \end{aligned} \tag{3.7}$$

where $a_1(t)$, defined as the *weighting function*, is the time-domain form of $A_1(\omega)$, and

$$\begin{aligned} f_k(t) &= 2v_k(t) - b_k(t), \\ f_m(t) &= 2v_m(t) - b_m(t). \end{aligned} \tag{3.8}$$

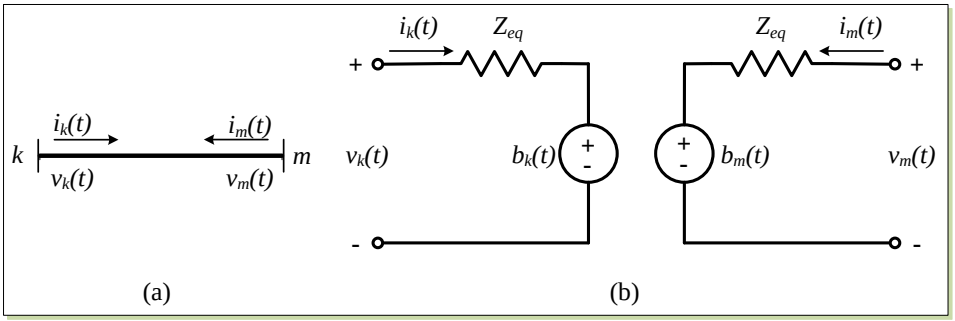


Figure 3.1: (a) A transmission line, and (b) its frequency-dependent model.

Accuracy of Marti's line model greatly depends on the fitting of $Z_c(\omega)$ and $A_1(\omega)$. The appropriate techniques used are Bode's asymptotic fitting technique [56] in the EMTP. $Z_c(\omega)$ is fitted by an N_{zc}-order rational function of the form

$$Z_{eq}(s) = k_0 + \sum_{i=1}^{N_{zc}} \frac{k_i}{s + p_i}, \tag{3.9}$$

which is realized by a series of RC parallel blocks as shown in Fig. 3.2 (a). The parameters are calculated as follows

$$R_0 = k_0,\ R_i = k_i/p_i,\ C_i = 1/k_i. \tag{3.10}$$

By applying Trapezoidal rule of integration to each RC parallel block its discrete-time model is obtained as shown in Fig. 3.2 (b), where

$$R_{eqi} = \frac{R_i \frac{\Delta t}{2C_i}}{R_i + \frac{\Delta t}{2C_i}}, \tag{3.11}$$

and

$$\begin{aligned} v_{hi}(t - \Delta t) &= \frac{R_i^2 \frac{\Delta t}{C_i}}{(R_i + \frac{\Delta t}{2C_i})^2} i(t - \Delta t) + \frac{R_i - \frac{\Delta t}{2C_i}}{R_i + \frac{\Delta t}{2C_i}} v_{hi}(t - 2\Delta t) \\ &= D_{1i} i(t - \Delta t) + D_{2i} v_{hi}(t - 2\Delta t). \end{aligned} \tag{3.12}$$

Combining all RC blocks discrete-time models together gives the overall Thévenin equivalent network shown in Fig. 3.2 (c), where

$$R_{eq} = R_0 + \sum_{i=1}^{N_{zc}} R_{eqi}, \tag{3.13}$$

and

$$v_h(t - \Delta t) = \sum_{i=1}^{N_{zc}} v_{hi}(t - \Delta t). \tag{3.14}$$

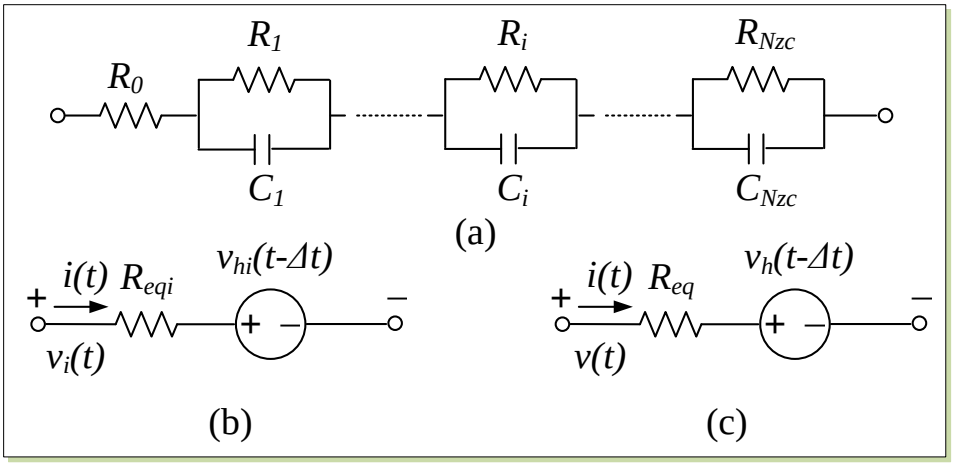


Figure 3.2: (a) RC network realization of $Z_{eq}(\omega)$ approximating $Z_c(\omega)$, (b) discrete-time model of i^{th} RC block, and (c) overall Thévenin equivalent network of $Z_{eq}(\omega)$.

Specifically, for sending-end ($'k'$) and receiving-end ($'m'$), (3.14) becomes

$$\begin{aligned} v_{hk}(t-\Delta t) &= \sum_{i=1}^{N_{zc}} v_{hki}(t-\Delta t) = \sum_{i=1}^{N_{zc}} D_{1i} i_k(t-\Delta t) + D_{2i} v_{hki}(t-2\Delta t), \\ v_{hm}(t-\Delta t) &= \sum_{i=1}^{N_{zc}} v_{hmi}(t-\Delta t) = \sum_{i=1}^{N_{zc}} D_{1i} i_m(t-\Delta t) + D_{2i} v_{hmi}(t-2\Delta t). \end{aligned} \tag{3.15}$$

Again, N_{zc} is total number of RC parallel blocks or the order of the approximation of $Z_c(\omega)$; i_k and i_m are line currents at both ends. Computational efficiency of the convolutions of (3.7) may be greatly increased if the weighting function $a_1(t)$ has the form of sum of exponential terms [55]. To do so, the same fitting techniques for $Z_c(\omega)$ are applied in the approximation of weighting function $A_1(\omega)$. However, rather than a proper form of $Z_{eq}(s)$, the $A_1(\omega)$ is first back-winded by the propagation delay τ to produce

$$P(\omega) = A_1(\omega) e^{j\omega\tau}, \tag{3.16}$$

and then approximated in a strictly proper manner by a N_p-order rational function

$$P_a(s) = \sum_{i=1}^{N_p} \frac{k_i}{s + p_i}. \tag{3.17}$$

Thus we obtain a sum of exponentials from the inverse Fourier transformation as

$$a_{1a}(t) = u(t-\tau) \sum_{i=1}^{N_p} k_i e^{-p_i(t-\tau)}, \tag{3.18}$$

where $u(t-\tau)$ is step response with τ delay.

For the i-th term $k_i e^{-p_i(t-\tau)}u(t-\tau)$ in approximated function $a_{1a}(t)$, the convolution integral is

$$b_i(t) = \int_{\tau}^{\infty} f(t-u)k_i e^{-p_i(u-\tau)}du. \tag{3.19}$$

$b_i(t)$ can then be directly obtained from recursive convolution [55] by

$$b_i(t) = M_i \cdot b_i(t-\Delta t) + P_i \cdot f(t-\tau) + Q_i \cdot f(t-\tau-\Delta t), \tag{3.20}$$

where

$$\begin{cases} M_i &= e^{-p_i\Delta t} = \alpha, \\ P_i &= \frac{k_i}{p_i}(1-\frac{1-\alpha}{p_i\Delta t}), \\ Q_i &= \frac{k_i}{p_i}(\frac{1-\alpha}{p_i\Delta t}-\alpha). \end{cases} \tag{3.21}$$

Specifically, $b_k(t)$ for sending-end ('k') and $b_m(t)$ for receiving-end ('m') are given as

$$\begin{aligned} b_k(t) &= \sum_{i=1}^{N_p} b_{ki}(t) = \sum_{i=1}^{N_p} [M_i \cdot b_{ki}(t-\Delta t) + P_i \cdot f_m(t-\tau) + Q_i \cdot f_m(t-\tau-\Delta t)], \\ b_m(t) &= \sum_{i=1}^{N_p} b_{mi}(t) = \sum_{i=1}^{N_p} [M_i \cdot b_{mi}(t-\Delta t) + P_i \cdot f_k(t-\tau) + Q_i \cdot f_k(t-\tau-\Delta t)]. \end{aligned} \tag{3.22}$$

To calculate (3.22), the past values of $f_m(t)$ and $f_k(t)$ have to be stored. The number of stored past value N_{tau} depends on the propagation delay τ and time-step Δt as follows

$$N_{tau} = \tau/\Delta t. \tag{3.23}$$

Finally, by combining (3.15) with (3.22), the equivalent time-domain network of FDLM model is obtained as shown in Fig. 3.3, where the equivalent current sources are given by

$$\begin{aligned} i_{hlnk}(t) &= [b_k(t) + v_{hk}(t-\Delta t)]/R_{eq}, \\ i_{hlnm}(t) &= [b_m(t) + v_{hm}(t-\Delta t)]/R_{eq}, \end{aligned} \tag{3.24}$$

and the equivalent resistance is given by (3.13).

The line currents for sending-end ('k') and receiving-end ('m') are calculated as

$$\begin{aligned} i_k(t) &= v_k(t)/R_{eq} - i_{hlnk}(t), \\ i_m(t) &= v_m(t)/R_{eq} - i_{hlnm}(t). \end{aligned} \tag{3.25}$$

Three-phase system is commonly decoupled into three separate systems. In the EMTP, the Clarke's transformation is used to transform the three phases to two aerial modes (α, β) and one ground mode (0) [5]. The Clarke's transformation matrix is defined as

$$\boldsymbol{i}_{phase} = \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \boldsymbol{T}\boldsymbol{i}_{mode} = \boldsymbol{T} \begin{bmatrix} i_0 \\ i_\alpha \\ i_\beta \end{bmatrix}, \tag{3.26}$$

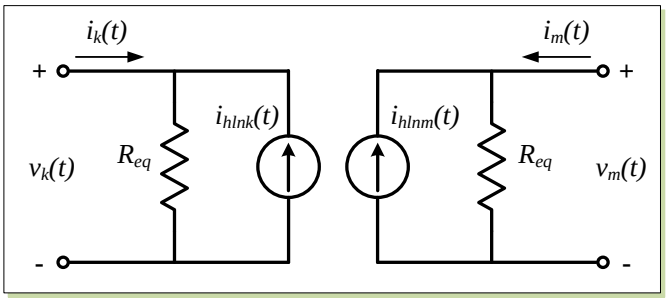


Figure 3.3: Discrete-time equivalent network for FDLM.

$$\boldsymbol{v}_{phase} = \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix} = \boldsymbol{T}\boldsymbol{v}_{mode} = \boldsymbol{T} \begin{bmatrix} v_0 \\ v_\alpha \\ v_\beta \end{bmatrix}, \tag{3.27}$$

where the transformation matrix $\mathbf{T}$ is given by

$$\boldsymbol{T} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & \sqrt{2} & 0 \\ 1 & -1/\sqrt{2} & \sqrt{3}/\sqrt{2} \\ 1 & -1/\sqrt{2} & -\sqrt{3}/\sqrt{2} \end{bmatrix}. \tag{3.28}$$

3.1.2 Real-Time FPGA Implementation of FDLM

A hardware module `FDLM` is designed to emulate FDLM model in the FPGA. Fig. 3.4 shows the symbol of the `FDLM` module and its input/output signals. The main operations carried out in this module include convolution for $b_k(t)$, $b_m(t)$ (3.22), calculating $i_{hlnk}(t)$, $i_{hlnm}(t)$ (3.24), and updating f_k, f_m (3.8) and $v_{hk}(t-\Delta t)$, $v_{hm}(t-\Delta t)$ (3.15).

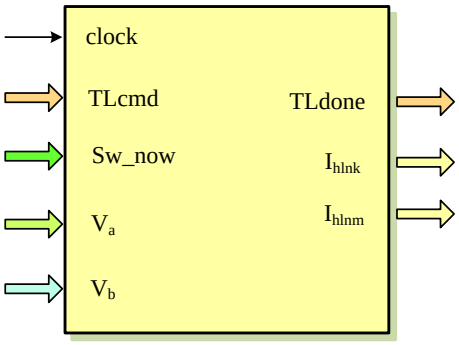


Figure 3.4: `FDLM` module and its input/output signals.

Due to the traveling time delay of FDLM model, the calculations at sending-end and receiving-end are naturally decoupled, which means they can be processed fully in parallel in hardware. Moreover, for three-phase systems, the calculations in α mode and β mode can also be carried out simultaneously. As can be seen in Fig. 3.5, four identical hardware

parts are implemented in the FPGA, and each part contains a `Convolution` unit and a `Update` unit.

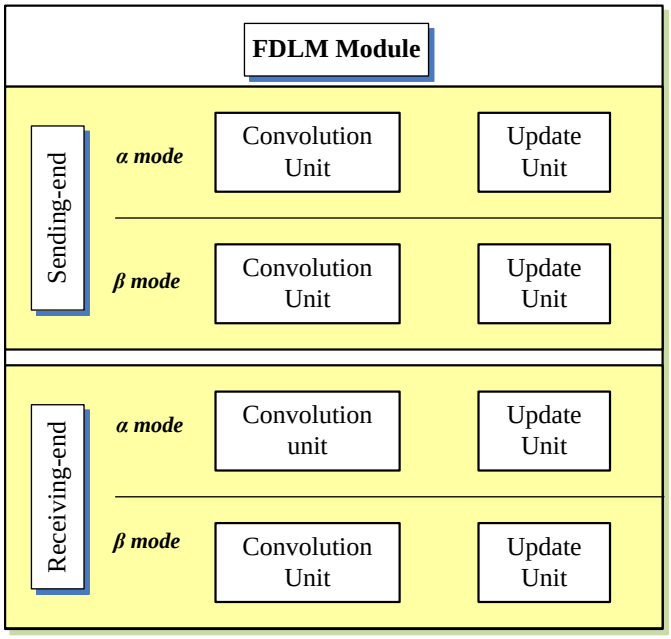


Figure 3.5: Functional units in the `FDLM` module showing parallel computations.

`Convolution` Unit

Depending on the order of fitted rational function N_p the recursive convolutions of (3.22) are very time-consuming in the EMTP. The `Convolution` unit is fully pipelined to improve the data throughput. Fig. 3.6 shows the pipelined computation scheme in this unit. A floating-point multiply-add ($y = a \times b + c \times d + e \times f$) unit is first used to calculate $b_{ki}(t)$, followed by a floating-point accumulator ($y = \sum a_i$) unit to calculate $b_k(t)$. Then the equivalent current $i_{hlnk}(t)$ (3.24) is calculated using another floating-point multiply-add ($y = a \times b + c \times d$) unit. RAMs are utilized to save the parameters such as M_i, P_i, and Q_i. The RAM used to save b_{ki} is a dual-port RAM which can be read out and written in at the same time.

`Update` Unit

After the network is solved, the history terms are updated in the `Update` unit shown in Fig. 3.7. The updated terms include $f_k(t)$, $f_m(t)$ (3.8) and $v_{hk}(t-\Delta t)$, $v_{hm}(t-\Delta t)$ (3.15). As can be seen in Fig. 3.7, a floating-point multiply-add ($y = a \times b - c \times d$) unit is employed to calculate f_k, while the other floating-point multiply-add ($y = a \times b - c \times d$) unit is employed to calculate line current i_k (3.25) concurrently. Then $v_{hk}(t - \Delta t)$ is calculated in a floating-

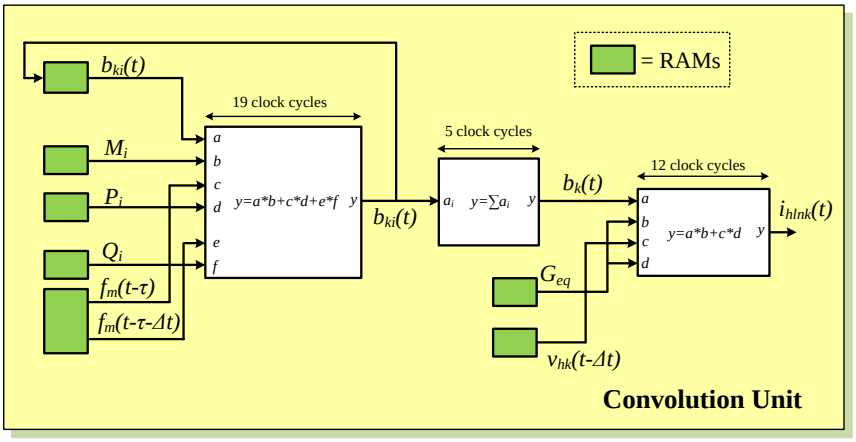


Figure 3.6: Pipelined computation scheme in the `Convolution` unit.

point multiply-add ($y = a \times b + c \times d$) unit and a floating-point accumulator ($y = \sum a_i$) unit.

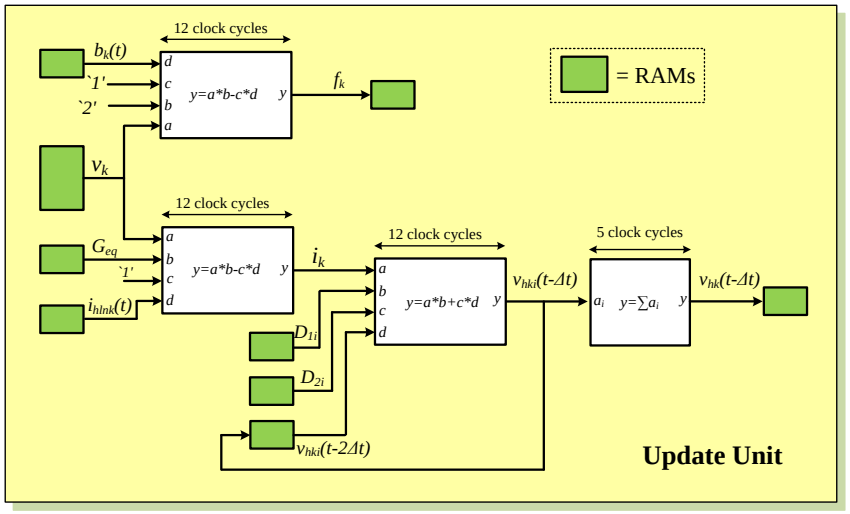


Figure 3.7: Pipelined computation scheme in the `Update` unit.

3.2 Linear Lumped RLCG Elements

3.2.1 Model Formulation

In the EMTP, the linear lumped elements R, L, C, and G are represented as their discrete-time models using trapezoidal rule of integration. Meanwhile, the series combinations of basic elements such as RL, RC, LC, RLC, $RLCG$ are treated as single branch element in order to reduce the number of nodes of network simulated.

Resistance R Element

The discrete-time model of a resistance R shown in Fig. 3.8 is the same as its continuous-time model. The voltage and current relationship is given as

$$v_{km}(t) = Ri_{km}(t). \tag{3.29}$$

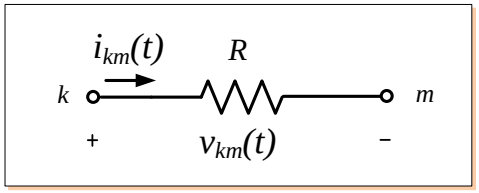


Figure 3.8: A resistance R element and its discrete-time model.

Inductance L Element

The differential equation for an inductance L shown in Fig. 3.9 (a) is

$$v_{km}(t) = L\frac{di_{km}(t)}{dt}. \tag{3.30}$$

Integrating this equation form time $(t - \Delta t)$ to t gives

$$i_{km}(t) - i_{km}(t - \Delta t) = \frac{1}{L}\int_{t-\Delta t}^{t} v_{km}(t)dt. \tag{3.31}$$

Applying the trapezoidal rule of integration yields

$$\begin{aligned} i_{km}(t) &= i_{km}(t - \Delta t) + \frac{\Delta t}{2L}[v_{km}(t) + v_{km}(t - \Delta t)] \\ &= \frac{\Delta t}{2L}v_{km}(t) + [i_{km}(t - \Delta t) + \frac{\Delta t}{2L}v_{km}(t - \Delta t)] \\ &= \frac{v_{km}(t)}{R_{eq}} + I_h(t - \Delta t). \end{aligned} \tag{3.32}$$

(3.32) can be represented by an equivalent resistance R_{eq} in parallel with a current source (history term) $I_h(t - \Delta t)$ as shown in Fig. 3.9 (b), where the equivalent resistance R_{eq} is defined as

$$R_{eq} = \frac{2L}{\Delta t}, \tag{3.33}$$

and the history term $I_h(t - \Delta t)$ is known from the solution at the preceding time-step and updated for use in the next time-step using

$$I_h(t - \Delta t) = [i_{km}(t - \Delta t) + \frac{\Delta t}{2L}v_{km}(t - \Delta t)], \tag{3.34}$$

or recursively using

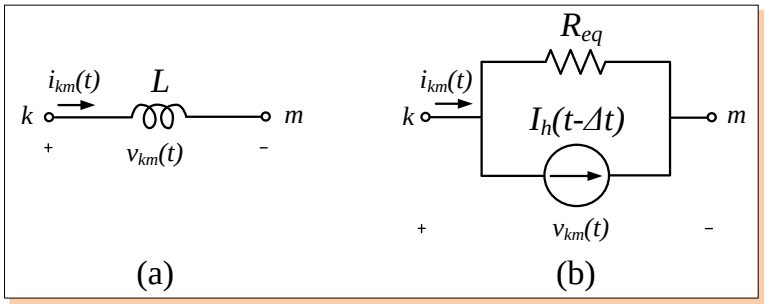


Figure 3.9: (a) An inductance L element, and (b) its discrete-time Norton equivalent.

$$\begin{aligned} I_h(t-\Delta t) &= I_h(t-2\Delta t) + \frac{\Delta t}{L} v_{km}(t-\Delta t) \\ &= I_h(t-2\Delta t) + \frac{2}{R_{eq}} v_{km}(t-\Delta t). \end{aligned} \tag{3.35}$$

Capacitance C Element

The differential equation for a capacitance C shown in Fig. 3.10 (a) is given as

$$i_{km}(t) = C\frac{dv_{km}(t)}{dt}. \tag{3.36}$$

Integrating this equation form time $(t-\Delta t)$ to t gives

$$v_{km}(t) - v_{km}(t-\Delta t) = \frac{1}{C}\int_{t-\Delta t}^{t} i_{km}(t)dt. \tag{3.37}$$

Applying the trapezoidal rule of integration yields

$$\begin{aligned} i_{km}(t) &= -i_{km}(t-\Delta t) + \frac{2C}{\Delta t}[v_{km}(t) - v_{km}(t-\Delta t)] \\ &= \frac{2C}{\Delta t} v_{km}(t) + [-i_{km}(t-\Delta t) - \frac{2C}{\Delta t} v_{km}(t-\Delta t)] \\ &= \frac{2C}{\Delta t} v_{km}(t) + I_h(t-\Delta t). \end{aligned} \tag{3.38}$$

(3.38) can be represented by an equivalent resistance R_{eq} in parallel with a current source (history term) $I_h(t-\Delta t)$ as shown in Fig. 3.10 (b), where the equivalent resistance R_{eq} is defined as

$$R_{eq} = \frac{\Delta t}{2C}, \tag{3.39}$$

and the history term $I_h(t-\Delta t)$ is known from the solution at the preceding time-step and updated for use in the next time-step using

$$I_h(t-\Delta t) = -i_{km}(t-\Delta t) - \frac{2C}{\Delta t} v_{km}(t-\Delta t), \tag{3.40}$$

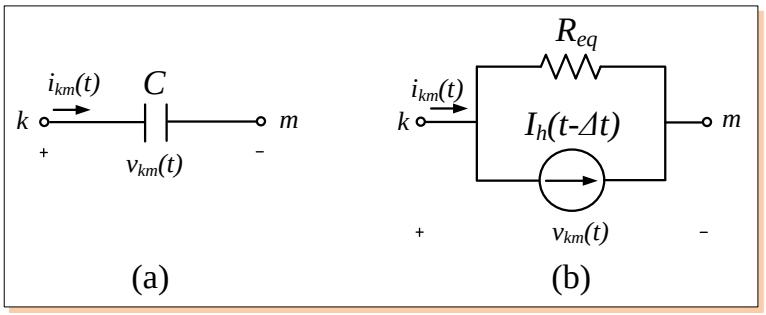


Figure 3.10: (a) A capacitance C element, and (b) its discrete-time Norton equivalent.

or recursively using

$$\begin{aligned} I_h(t-\Delta t) &= -I_h(t-2\Delta t) - \frac{4C}{\Delta t} v_{km}(t-\Delta t) \\ &= -I_h(t-2\Delta t) - \frac{2}{R_{eq}} v_{km}(t-\Delta t). \end{aligned} \tag{3.41}$$

Series RL Branch Element

The series RL branch element shown in Fig. 3.11 (a) is first replaced by the discrete-time model of R and L shown in Fig. 3.11 (b).

Combining (3.29) and (3.32) gives

$$v_{km}(t) = R i_{km}(t) + [i_{km}(t) - i_{km}(t-\Delta t) - \frac{\Delta t}{2L} v_L(t-\Delta t)] \frac{2L}{\Delta t}. \tag{3.42}$$

Substituting $v_L(t-\Delta t)$ and rearranging yields

$$i_{km}(t) = \frac{1}{R + \frac{2L}{\Delta t}} v_{km}(t) + \frac{1}{R + \frac{2L}{\Delta t}} [v_{km}(t-\Delta t) - (R - \frac{2L}{\Delta t}) i_{km}(t-\Delta t)]. \tag{3.43}$$

(3.43) can be represented by an equivalent resistance R_{eq} in parallel with a current source (history term) $I_h(t-\Delta t)$ as shown in Fig. 3.11 (c), where the equivalent resistance R_{eq} is defined as

$$R_{eq} = R + \frac{2L}{\Delta t}, \tag{3.44}$$

and the history term $I_h(t-\Delta t)$ is known from the solution at the preceding time-step and updated for use in the next time-step using

$$I_h(t-\Delta t) = \frac{v_{km}(t-\Delta t)}{R + \frac{2L}{\Delta t}} - \frac{(R - \frac{2L}{\Delta t}) i_{km}(t-\Delta t)}{R + \frac{2L}{\Delta t}}, \tag{3.45}$$

or recursively using

$$I_h(t-\Delta t) = -\frac{R - \frac{2L}{\Delta t}}{R_{eq}} I_h(t-2\Delta t) + \frac{\frac{4L}{\Delta t}}{R_{eq}^2} v_{km}(t-\Delta t). \tag{3.46}$$

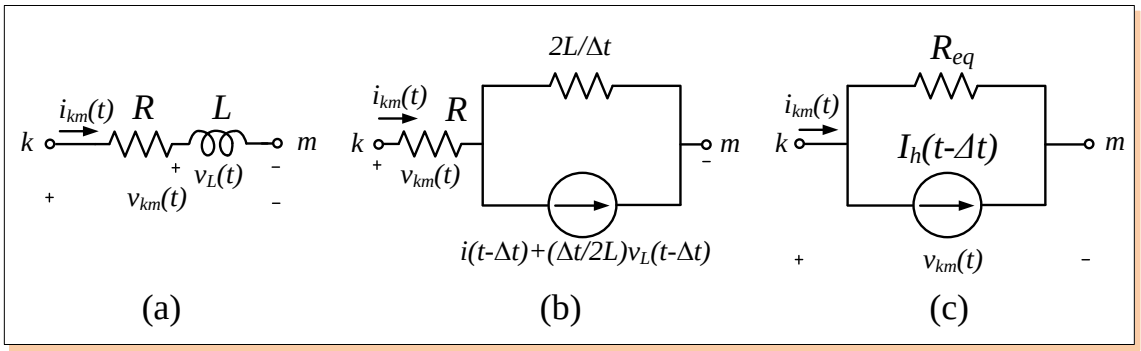


Figure 3.11: (a) A series RL branch element, (b) combined R, L discrete-time models, and (c) its Norton equivalent.

Series RC Branch Element

Similarly, combining the discrete-time model of R and C, a series RC branch element shown in Fig. 3.12 (a) can be represented as its equivalent network shown in Fig. 3.12 (b), where the equivalent resistance R_{eq} is defined as

$$R_{eq} = R + \frac{\Delta t}{2C}, \tag{3.47}$$

and the history term $I_h(t - \Delta t)$ is known from the solution at the preceding time-step and updated recursively for use in the next time-step using

$$I_h(t - \Delta t) = -\frac{R - \frac{\Delta t}{2C}}{R_{eq}} I_h(t - 2\Delta t) - \frac{2R}{R_{eq}^2} v_{km}(t - \Delta t). \tag{3.48}$$

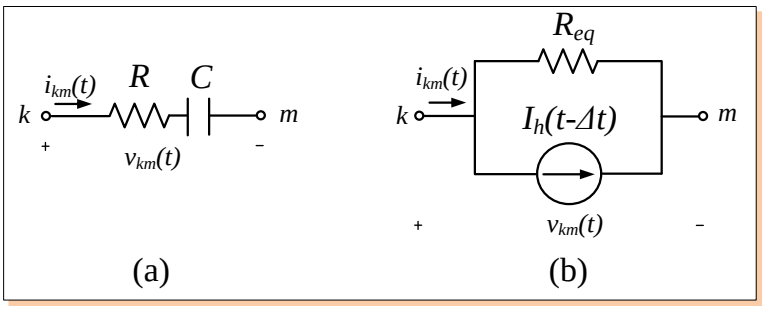


Figure 3.12: (a) A series RC branch element, and (b) its discrete-time Norton equivalent.

Series LC Branch Element

Combining the discrete-time model of L and C, a series LC branch element shown in Fig. 3.13 (a) can be represented as its equivalent network shown in Fig. 3.13 (b), where the

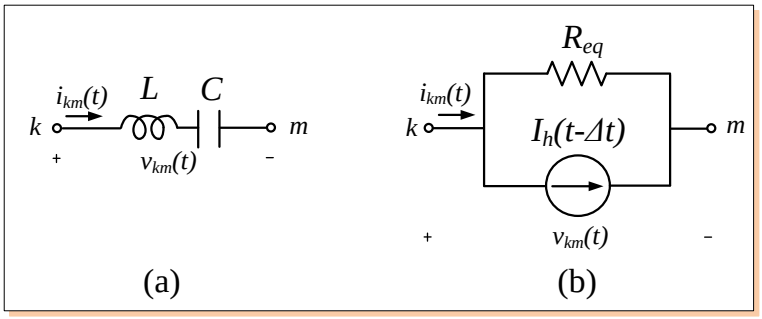


Figure 3.13: (a) A series LC branch element ,and (b) its discrete-time Norton equivalent.

equivalent resistance R_{eq} is defined as

$$R_{eq} = \frac{2L}{\Delta t} + \frac{\Delta t}{2C}, \tag{3.49}$$

and the history term $I_h(t - \Delta t)$ is known from the solution at the preceding time-step and updated recursively for use in the next time-step using

$$\begin{cases} I_h(t-\Delta t) = \frac{1}{R_{eq}}[V_{h,L}(t-\Delta t) + V_{h,C}(t-\Delta t)], \\ V_{h,L}(t-\Delta t) = -V_{h,L}(t-2\Delta t) + \frac{4L}{\Delta t} i_{km}(t-\Delta t), \\ V_{h,C}(t-\Delta t) = V_{h,C}(t-2\Delta t) - \frac{\Delta t}{C} i_{km}(t-\Delta t), \\ i_{km}(t-\Delta t) = \frac{1}{R_{eq}} v_{km}(t-\Delta t) + I_h(t-\Delta t). \end{cases} \tag{3.50}$$

Series RLC Branch Element

Combining the discrete-time model of R, L and C, a series RLC branch element shown in Fig. 3.14 (a) can be represented as its equivalent network shown in Fig. 3.14 (b), where the equivalent resistance R_{eq} is defined as

$$R_{eq} = R + \frac{2L}{\Delta t} + \frac{\Delta t}{2C}, \tag{3.51}$$

and the history term $I_h(t - \Delta t)$ is known from the solution at the preceding time-step and updated recursively for use in the next time-step using

$$\begin{cases} I_h(t-\Delta t) = \frac{1}{R_{eq}}[V_{h,L}(t-\Delta t) + V_{h,C}(t-\Delta t)], \\ V_{h,L}(t-\Delta t) = -V_{h,L}(t-2\Delta t) + \frac{4L}{\Delta t} i_{km}(t-\Delta t), \\ V_{h,C}(t-\Delta t) = V_{h,C}(t-2\Delta t) - \frac{\Delta t}{C} i_{km}(t-\Delta t), \\ i_{km}(t-\Delta t) = \frac{1}{R_{eq}} v_{km}(t-\Delta t) + I_h(t-\Delta t). \end{cases} \tag{3.52}$$

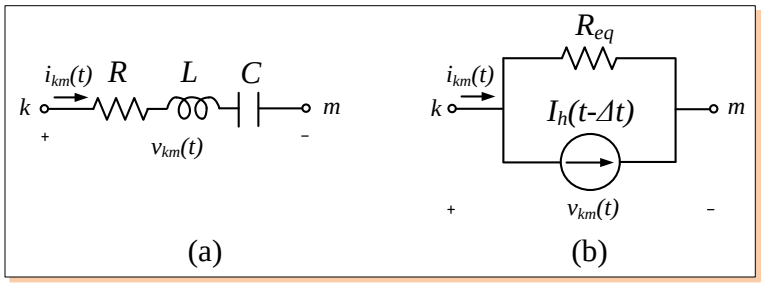


Figure 3.14: (a) A series RLC branch element, and (b) its discrete-time Norton equivalent.

$RLCG$ Branch

Combining the discrete-time model of R, L, C, and G, the $RLCG$ element shown in branch Fig. 3.15 (a) can be represented as its equivalent network shown in Fig. 3.15 (b), where the equivalent resistance R_{eq} is defined as

$$R_{eq} = R + \frac{2L}{\Delta t} + \frac{G\frac{\Delta t}{2C}}{G + \frac{\Delta t}{2C}}, \tag{3.53}$$

and the history term $I_h(t - \Delta t)$ is given as

$$\begin{cases} I_h(t-\Delta t) = \frac{1}{R_{eq}}[V_{h,L}(t-\Delta t) + V_{h,CG}(t-\Delta t)], \\ V_{h,L}(t-\Delta t) = -V_{h,L}(t-2\Delta t) + \frac{4L}{\Delta t} i_{km}(t-\Delta t), \\ V_{h,CG}(t-\Delta t) = \frac{G-\frac{\Delta t}{2C}}{G+\frac{\Delta t}{2C}} V_{h,CG}(t-2\Delta t) - \frac{G^2\frac{\Delta t}{C}}{(G+\frac{\Delta t}{2C})^2} i_{km}(t-\Delta t), \\ i_{km}(t-\Delta t) = \frac{1}{R_{eq}} v_{km}(t-\Delta t) + I_h(t-\Delta t). \end{cases} \tag{3.54}$$

Note that G is in ohms.

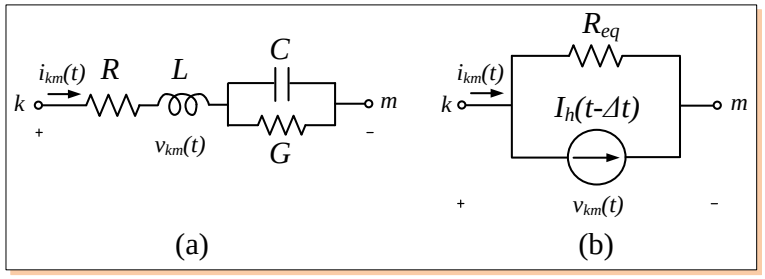


Figure 3.15: (a) $RLCG$ branch, and (b) its discrete-time Norton equivalent.

It is clear from the above discussion that all the RLCG elements share the same discrete-time equivalent network which is an equivalent conductance G_{eq} in parallel with a history current $I_h(t-\Delta t)$. Moreover, the history current update equations (3.35), (3.41), (3.46), and (3.48) have the same format, while equations (3.50), (3.52), and (3.54) have the same format.

Thus, all the RLCG elements can be divided into two types: `RLCGtype1` which includes R, L, C, RL, and RC elements and `RLCGtype2` which includes LC, RLC, and $RLCG$ elements. Although it is possible to have a general formula to include all types of RLCG elements, the number of `RLCGtype1` RLCG elements in power systems is higher than the number of `RLCGtype2` elements. The formula of updating history current of `RLCGtype1` is much simpler than that of `RLCGtype2`, which will improve the computational efficiency significantly.

The generic equation for updating `RLCGtype1` elements history currents is

$$I_{hpe1}(t-\Delta t) = P_1 I_{hpe1}(t-2\Delta t) + P_2 v_{km}(t-\Delta t), \tag{3.55}$$

where P_1, P_2 are coefficients with respect to different elements which are listed in Table 3.1. Note for resistance R, P_1, P_2 are zeros since there is no equivalent current source.

Table 3.1: Coefficients for updating `RLCGtype1` elements history currents (3.55)

`RLCGtype1` elements	$R_{eq}(1/G_{eq})$	P_1	P_2
R	R	0	0
L	$\frac{2L}{\Delta t}$	1	$\frac{\Delta t}{L}$
C	$\frac{\Delta t}{2C}$	-1	$-\frac{4C}{\Delta t}$
RL	$R+\frac{2L}{\Delta t}$	$-\frac{R-\frac{2L}{\Delta t}}{R+\frac{2L}{\Delta t}}$	$\frac{\frac{4L}{\Delta t}}{(R+\frac{2L}{\Delta t})^2}$
RC	$R+\frac{\Delta t}{2C}$	$-\frac{R-\frac{\Delta t}{2C}}{R+\frac{\Delta t}{2C}}$	$-\frac{2R}{(R+\frac{\Delta t}{2C})^2}$

The generic equations for updating `RLCGtype2` elements history terms are given as

$$\begin{cases} I_{hpe2}(t-\Delta t) = \frac{1}{R_{eq}}[V_{h,L}(t-\Delta t) + V_{h,c}(t-\Delta t)], \\ V_{h,L}(t-\Delta t) = P_1 V_{h,L}(t-2\Delta t) + P_2 i_{km}(t-\Delta t), \\ V_{h,C}(t-\Delta t) = P_3 V_{h,C}(t-2\Delta t) + P_4 i_{km}(t-\Delta t), \end{cases} \tag{3.56}$$

where $R_{eq}(1/G_{eq})$, P_1, P_2, P_3, and P_4 are coefficients with respect to different elements which are listed in Table 3.2.

Table 3.2: Coefficients for updating `RLCGtype2` elements history currents (3.56)

`RLCGtype2` elements	$R_{eq}(1/G_{eq})$	P_1	P_2	P_3	P_4
LC	$\frac{2L}{\Delta t}+\frac{\Delta t}{2C}$	-1	$\frac{4L}{\Delta t}$	1	$-\frac{\Delta t}{C}$
RLC	$R+\frac{2L}{\Delta t}+\frac{\Delta t}{2C}$	-1	$\frac{4L}{\Delta t}$	1	$-\frac{\Delta t}{C}$
RLCG	$R+\frac{2L}{\Delta t}+\frac{G\frac{\Delta t}{2C}}{G+\frac{\Delta t}{2C}}$	-1	$\frac{4L}{\Delta t}$	$\frac{G-\frac{\Delta t}{2C}}{G+\frac{\Delta t}{2C}}$	$-\frac{G^2\frac{\Delta t}{C}}{(G+\frac{\Delta t}{2C})^2}$

The branch current for all RLCG elements is

$$i_{km}(t) = \frac{1}{R_{eq}} v_{km}(t) + I_{hpe}(t-\Delta t). \tag{3.57}$$

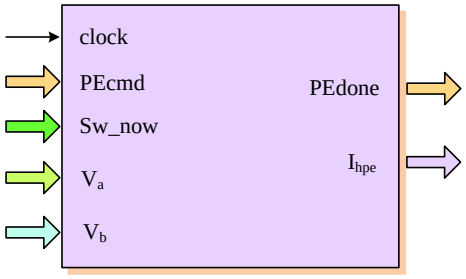


Figure 3.16: `RLCG` module and its input/output signals.

3.2.2 Real-Time FPGA Implementation of Linear Lumped RLCG Elements

A hardware module `RLCG` is designed to emulate the linear lumped RLCG elements in the FPGA. Fig. 3.16 shows the symbol of the `RLCG` module and its input/output signals. The main function of the `RLCG` module is to update history currents $I_{hpe}(t-\Delta t)$. The pipelined hardware computation schemes for updating the history currents for `RLCGtype1` (3.55) and `RLCGtype2` (3.56) RLCG elements are shown in Fig. 3.17 and Fig. 3.18, respectively.

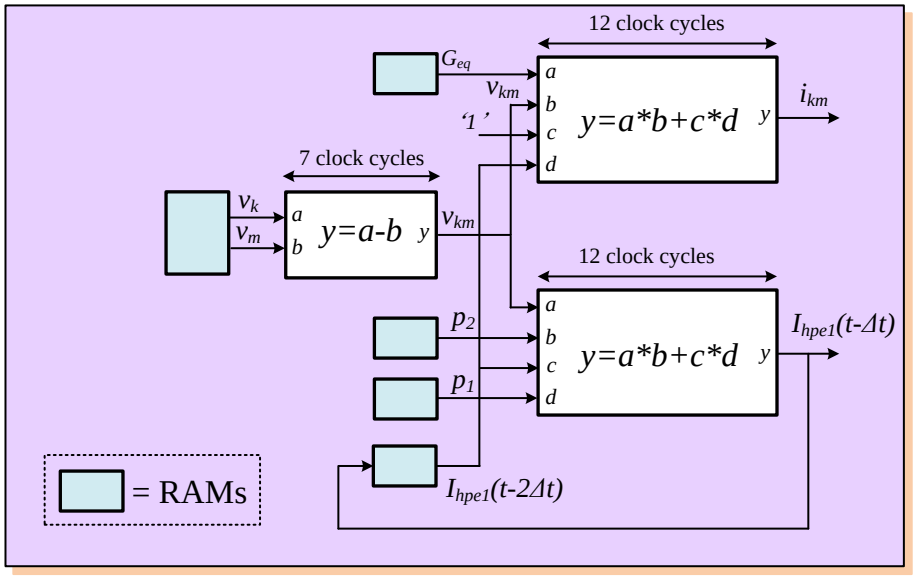


Figure 3.17: Pipelined computation scheme for calculating I_{hpe1} of `RLCGtype1` elements.

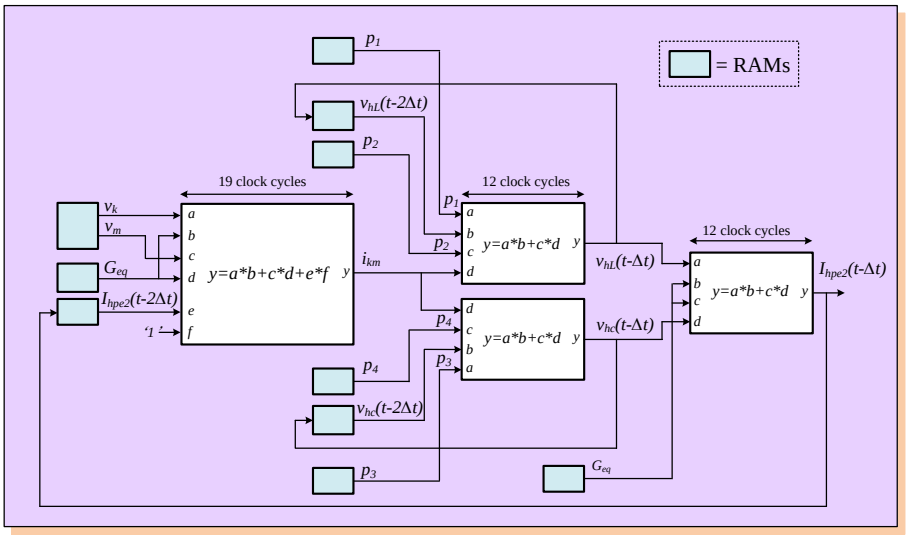


Figure 3.18: Pipelined computation scheme for calculating I_{hpe2} of `RLCGtype2` elements.

3.3 Sources

3.3.1 Modeling of Sources

The ideal voltage source and current source are represented using sinusoidal function (cos function) in the EMTP. The look-up table (LUT) is the most commonly used method to evaluate this nonlinear function. Since cos is a periodic function, as shown in Fig. 3.19 (a), only half cycle of cos function values need to be stored in the LUT in order to save the memory space of the LUT. The accuracy of the cos value is determined by the length of the LUT. In this design, 4096 (2^{12}) cos values for half cycle are stored in the LUT; thus, the interval of the LUT is $d\theta$=π/4096. Fig. 3.19 (b) shows the structure of the LUT.

A voltage or current source is expressed as

$$\begin{aligned} v(t), i(t) &= mag \cdot cos(\omega t + pha) \\ &= mag \cdot cos(\omega n \Delta t + pha), \end{aligned} \tag{3.58}$$

where mag, ω, and pha (0 to π) are the source magnitude, frequency, and phase angle, respectively. To retrieve the cos value from the LUT, the phase angle pha is converted to the address (0 to 4095) of the LUT.

3.3.2 Real-Time FPGA Implementation of Sources

A hardware module `Source` is designed to emulate the source components in the FPGA. Fig. 3.20 shows the symbol of the `Source` module and its input/output signals. The main function of the `Source` module is to calculate the source values at each simulation time-step.

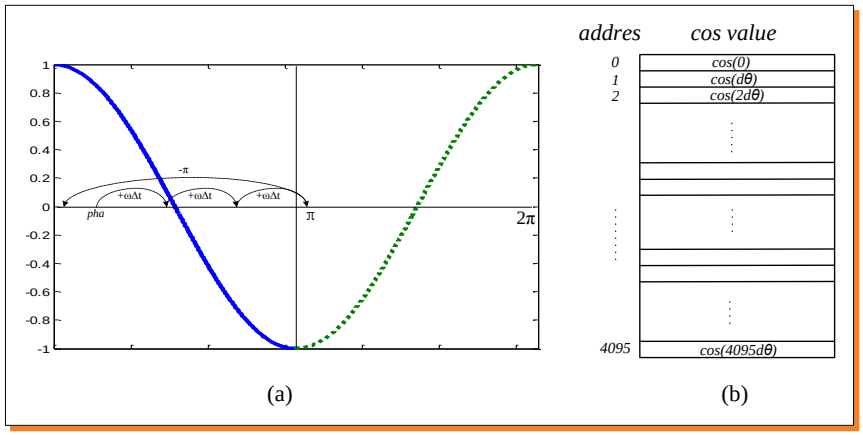


Figure 3.19: (a) *cos* function, and (b) structure of *cos* function look-up table.

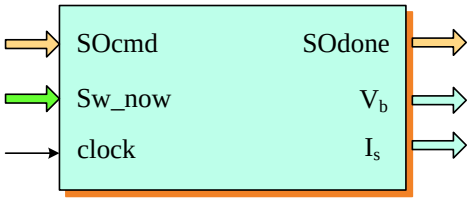


Figure 3.20: `Source` module and its input/output signals.

As shown in Fig. 3.21, the calculation of the source values begins with the updating of the phase angle pha. The new pha is obtained by adding the previous pha by $\omega\Delta t$. Since the LUT has only half cycle of cos functions values, the calculated pha needs to be checked with π; if greater than π, it is subtracted by π and the sign is inverted. Then the new pha is converted to the address of the LUT. Finally the retrieved cos value is multiplied by the magnitude mag. The type of the source (current or voltage) is checked before the calculated source values being saved as v_b or i_s.

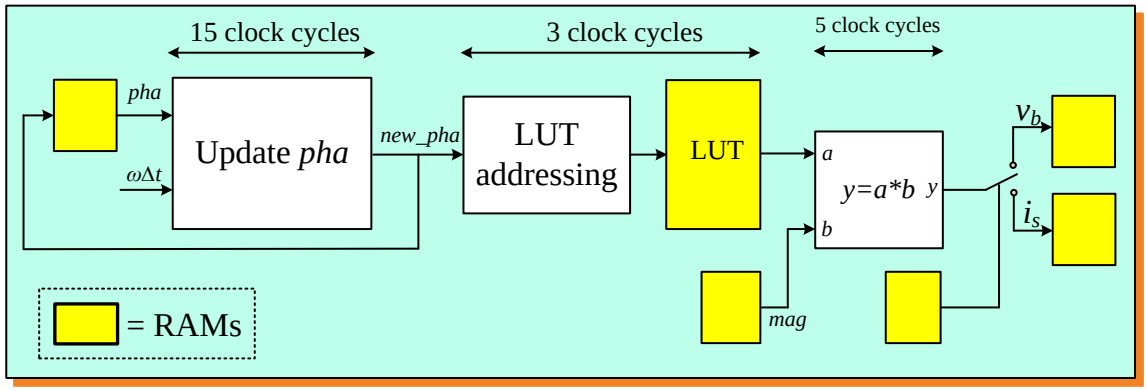


Figure 3.21: Pipelined configuration for calculating source values.

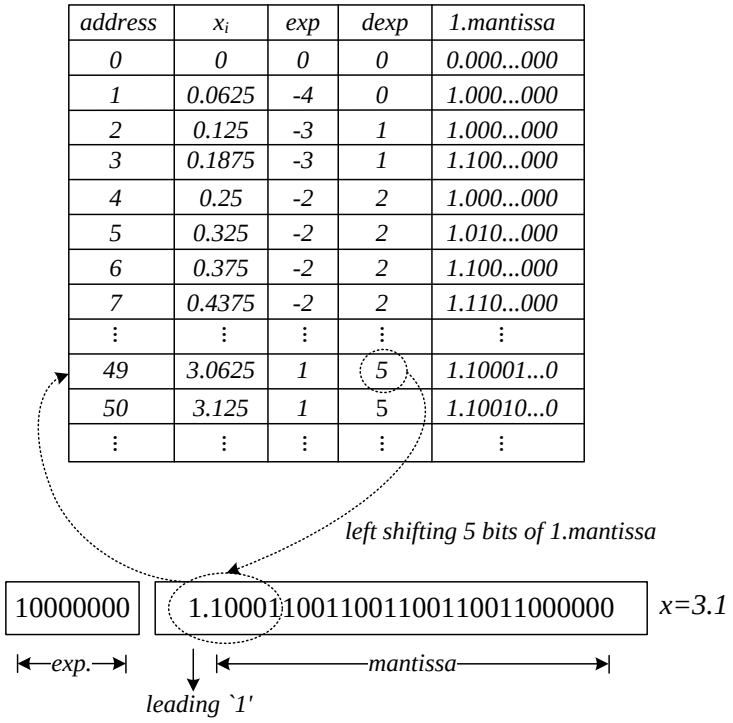


Figure 3.22: An example showing the scheme of `LUT addressing` unit.

One challenge of implementing a floating-point LUT is that the LUT is addressed by integer number whereas the input is in floating-point format. The straightforward method is to convert the floating-point input into integer format at the expense of extra latency for the conversion. In this design, this conversion is omitted. The exponent and mantissa of the input floating-point number are used directly to access the LUT provided the interval of the LUT is always a power of 2. Assume the floating-point input is x, the `LUT addressing` unit in Fig. 3.21 outputs the addresses of the point x_i making $x - x_i < step$, where $step$ is the interval of the LUT. This is done by left shifting the leading '1' and mantissa of x by $dexp$ bits, where the $dexp$ is the difference of the exponent of input x and $step$. An example is shown in Fig. 3.22. In this example, $step$ is 0.0625 (2^{-4}), and input x is 3.10 whose exponent is 1 (without bias). So $dexp$ is 5 (=$1 - (-4)$). Left shifting the leading '1' and mantissa 5 bits gives 49 which is the address of x_i (3.0625).

3.4 Circuit Breakers

3.4.1 Modeling of Circuit Breakers

In the EMTP, circuit breakers are commonly represented as ideal time-controlled switches shown in Fig. 3.23. It opens at t_{op} with infinite resistance (R=∞) and closes at t_{cl} with zero resistance (R=0). The operation of switch usually causes the change of the system topology leading to the change of network admittance matrix $\boldsymbol{Y}$.

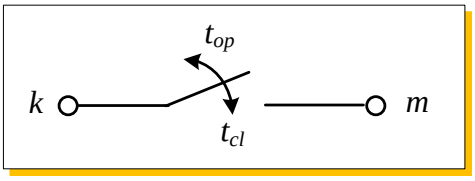


Figure 3.23: Ideal time-controlled switch.

3.4.2 Real-Time FPGA Implementation of Circuit Breakers

A hardware module `Switch` is designed to simulate circuit breakers in the FPGA. Fig. 3.24 shows the symbol of this module and its input/output signals. Since the switch is time-controlled, the core of this module is a real-time clock generator. As shown in Fig. 3.25, the input system clock frequency (80MHz) is divided by 800 to generate a $10\mu s$-period clock signal. This clock signal is counted and compared with the switch operation time $swtime$ (integer of $10\mu s$) saved in a RAM. Once the switch's $swtime$ is reached, the corresponding switch state bit in Sw_now register is inverted using '0'/ '1' for switch open/closed.

Figure 3.24: `Switch` module and its input/output signals.

Another important function of the `Switch` module is to generate the $dTover$ signal which indicates the end of the real simulation time-step Δt. When a simulation step is finished, the $dTover$ signal is checked. If it is not '1', the simulation step is finished within Δt, thus the real-time simulation is achieved. Otherwise, the simulation step takes longer time than Δt, and the real-time constrain is not met.

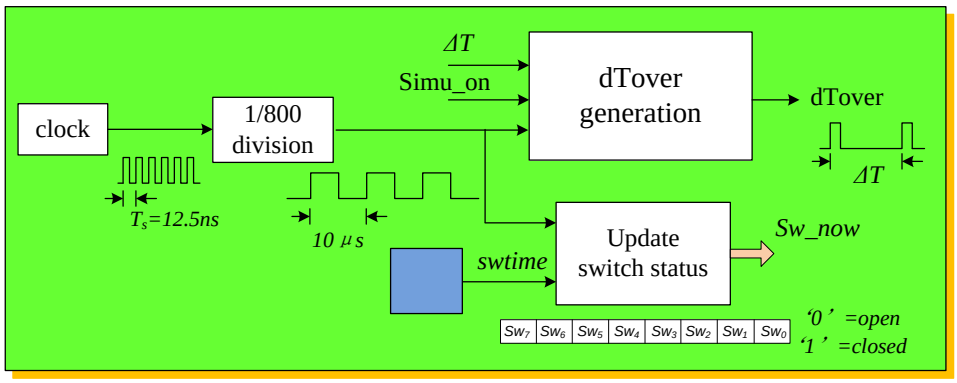


Figure 3.25: Functions realized in the `Switch` module.

3.5 Network Solver

3.5.1 Network Solution in the EMTP

After the discrete-time models of transmission lines and linear lumped RLCG elements are obtained, the system admittance matrix $\boldsymbol{Y}$ is assembled using nodal analysis method. $\boldsymbol{Y}$ is symmetrical and unchanged if there is no switch operation causing the change of system topology. For any type of network with n nodes, the nodal equation can be formed as

$$\boldsymbol{Y}\boldsymbol{v}(t) = \boldsymbol{i}(t), \tag{3.59}$$

where $\boldsymbol{v}(t)$ is vector of n node voltages and $\boldsymbol{i}(t)$ is vector of n current sources. If there are nodes with known voltages, (3.59) can be partitioned into a set A of nodes with unknown voltages and a set B of nodes with known voltages. Thus (3.59) becomes

$$\begin{bmatrix} \boldsymbol{Y}_{AA} & \boldsymbol{Y}_{AB} \\ \boldsymbol{Y}_{BA} & \boldsymbol{Y}_{BB} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_A \\ \boldsymbol{v}_B \end{bmatrix} = \begin{bmatrix} \boldsymbol{i}_A \\ \boldsymbol{i}_B \end{bmatrix}. \tag{3.60}$$

The unknown voltages $\boldsymbol{v}_A$ are then found by solving the system of linear, algebraic equations

$$\boldsymbol{Y}_{AA}\boldsymbol{v}_A = \boldsymbol{i}_A - \boldsymbol{i}_{A1}, \tag{3.61}$$

where

$$\boldsymbol{i}_{A1} = \boldsymbol{Y}_{AB}\boldsymbol{v}_B, \tag{3.62}$$

where $\boldsymbol{i}_A$ is the current sources of unknown nodes. It is assembled by transmission line history currents $\boldsymbol{i}_{hln}$, lumped RLCG element history currents $\boldsymbol{i}_{hpe}$, and known current sources $\boldsymbol{i}_s$ as follows

$$\boldsymbol{i}_A = \boldsymbol{i}_{hln} + \boldsymbol{i}_{hpe} + \boldsymbol{i}_s. \tag{3.63}$$

In the off-line EMTP, (3.61) is solved first by triangularizing the $\boldsymbol{Y}_{AA}$ matrix through a Gaussian elimination procedure and then backsubstituting for the updated values. In the real-time EMT simulator, (3.61) is solved by multiplying the precalculated inverse system admittance matrix $\boldsymbol{Y}_{AA}^{-1}$ by $(\boldsymbol{i}_A - \boldsymbol{i}_{A1})$. To account for switches in the network for a specific transient study, $\boldsymbol{Y}^{-1}$ corresponding to those switch combinations are precalculated and stored in the memory. Meanwhile, since $\boldsymbol{Y}$ and $\boldsymbol{Y}^{-1}$ matrices are very sparse for power systems, sparse matrix technique is employed for the matrix-vector multiplication, which reduces memory utilization and increases the calculation efficiency significantly. Fig. 3.26 shows the inverse admittance matrix for a modified IEEE 39-bus test system (Fig. 6.4). A 95.92% sparsity can be seen.

3.5.2 Real-Time FPGA Implementation of Network Solver

A hardware module `Network Solver` is designed to emulate the network solver in the FPGA. Fig. 3.27 shows the symbol of this module and its input/output signals. The main functions of this module are to calculate $\boldsymbol{i}_{A1}$ (3.62), $\boldsymbol{i}_A$ (3.63), and $\boldsymbol{v}_A$ (3.61).

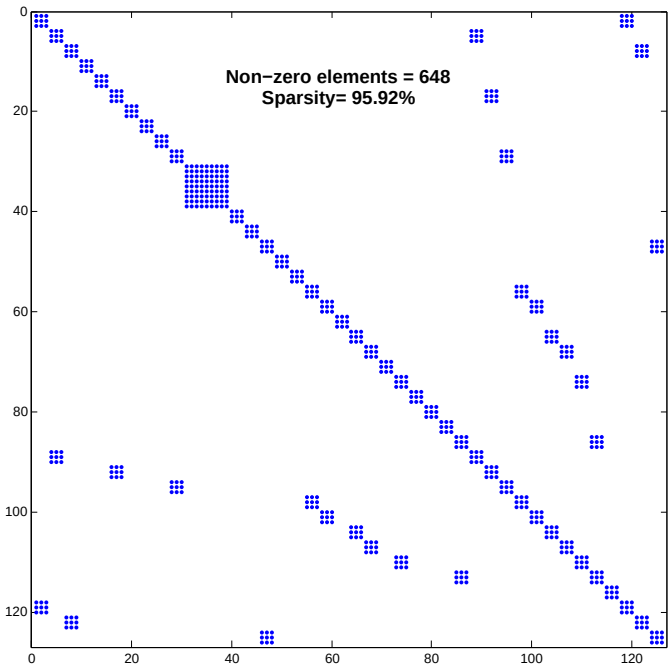


Figure 3.26: Inverse admittance matrix ($\boldsymbol{Y}^{-1}$) (126 x 126) of a modified IEEE 39-bus test system.

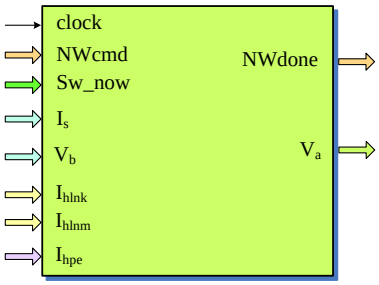


Figure 3.27: `Network Solver` module and its input/output signals.

As shown in Fig. 3.28, the calculation of $\boldsymbol{i}_{A1}$ and $\boldsymbol{i}_A$ are carried out concurrently. Meanwhile, $\boldsymbol{i} = \boldsymbol{i}_A - \boldsymbol{i}_{A1}$ is also calculated in order to calculate $\boldsymbol{v}_A$ shown in Fig. 3.29. As can be seen in these two figures, the core unit is a fast floating-point multiply-accumulator (`FFPMAC`) unit along with a compact sparse matrix storage format, which are discussed in detail as follows.

To carry out fast sparse matrix-vector multiplication, first a very compact sparse matrix storage format which uses only one vector is defined. Each entry in this format has (a) a 32-bit val to store the subsequent non-zero value of matrix in row order; (b) a 10-bit cid to identify column index of this non-zero value; and (c) a 1-bit rlb to label all non-zero values in the same row with $'0'$ or $'1'$. Fig. 3.30 shows an example sparse matrix and its storage format.

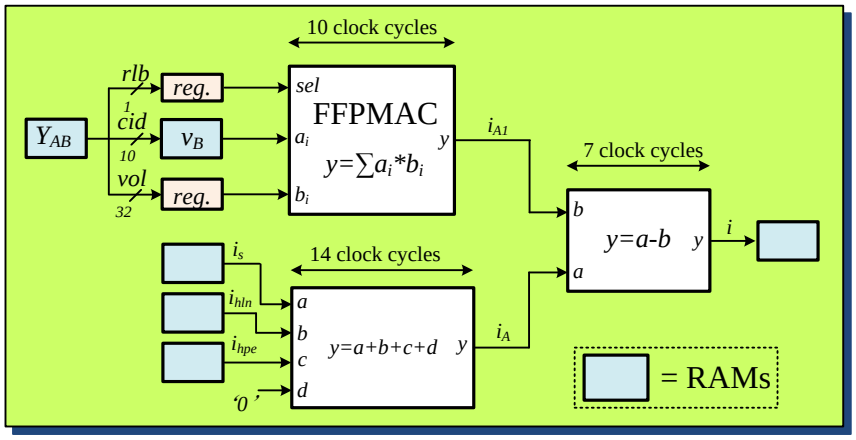


Figure 3.28: Pipelined and parallelled calculation scheme for $\boldsymbol{i}_{A1}$ and $\boldsymbol{i}_A$.

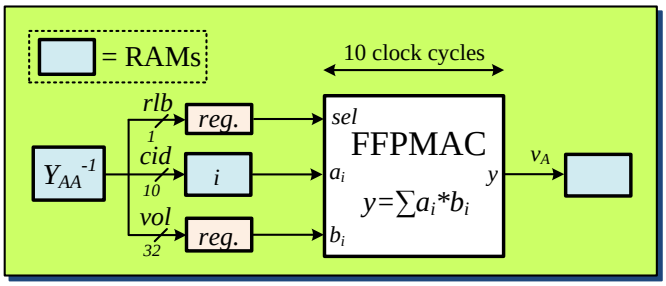


Figure 3.29: Pipelined calculation scheme for $\boldsymbol{v}_A$.

$$\begin{bmatrix} y_{00} & y_{01} & 0 & 0 & y_{04} \\ 0 & y_{11} & 0 & y_{13} & 0 \\ 0 & 0 & y_{22} & 0 & y_{24} \\ 0 & 0 & y_{32} & 0 & 0 \\ y_{40} & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a)

rlb	cid	val
0	0	y_{00}
0	1	y_{01}
0	4	y_{04}
1	1	y_{11}
1	3	y_{13}
0	2	y_{22}
0	4	y_{24}
1	2	y_{32}
0	0	y_{40}

(b)

Figure 3.30: (a) An example sparse matrix, and (b) its storage format.

In the `FFPMAC` unit shown in Fig. 3.31 (a), the accumulation is done in fixed-point format. The reason has been discussed in Section (2.3.1). Thus the `FFPMAC` unit contains one floating-point multiplier (`FPMUL`), one floating-to-fixed point converter (`FLP2FIX`), two fixed-point adders, and one fixed-to-floating point converter (`FIX2FLP`). To calculate $\boldsymbol{Y}_{AA}^{-1}\boldsymbol{i}$ shown in Fig. 3.29, for example, the elements of sparse matrix $\boldsymbol{Y}_{AA}^{-1}$ are retrieved from its RAM while the *cid* is used to access the $\boldsymbol{i}$ stored in its RAM. Registers are inserted

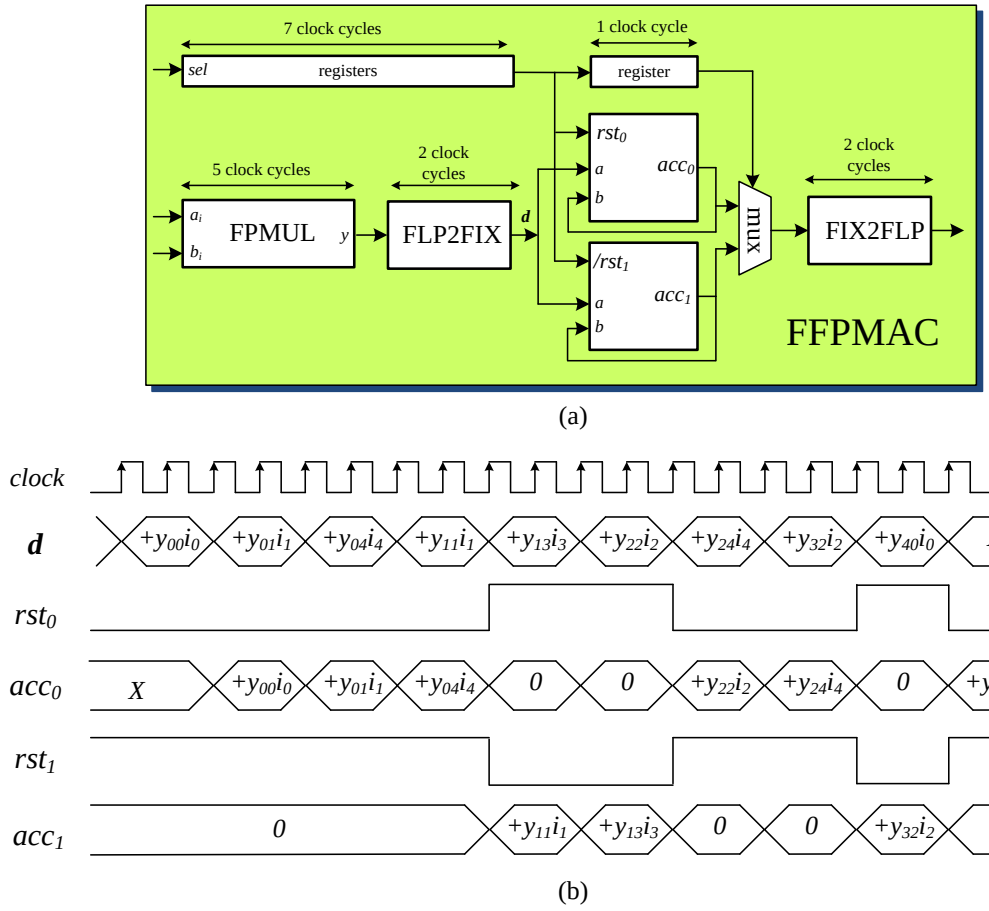


Figure 3.31: Fast floating-point multiply-accumulator unit (FFPMAC): (a) hardware design, and (b) timing diagram.

for synchronizing the pipeline. The realized matrix-vector multiplication is fully pipelined and fast because there is no stall between two consecutive matrix row-vector multiplications. This is achieved by two parallel fixed-point adders (accumulators) acc_0 and acc_1 with opposite reset inputs rst_0 and rst_1 controlled by the row label bit rlb. Fig. 3.31 (b) shows the logic timing diagram for the fast floating-point multiply-accumulator unit based on the example in Fig. 3.30. As can be seen, the accumulation of the first matrix row-vector multiplication is processed in acc_0 , while the acc_1 is reset to zero which makes it ready for accumulation of the next matrix row-vector multiplication.

3.6 Paralleled EMTP Algorithm

3.6.1 Analysis of Parallelism in the EMTP Algorithm

To fully exploit the parallel architecture of the FPGA, the inherent parallelism in the EMTP algorithm needs to be analyzed. The EMTP algorithm actually consists of three stages:

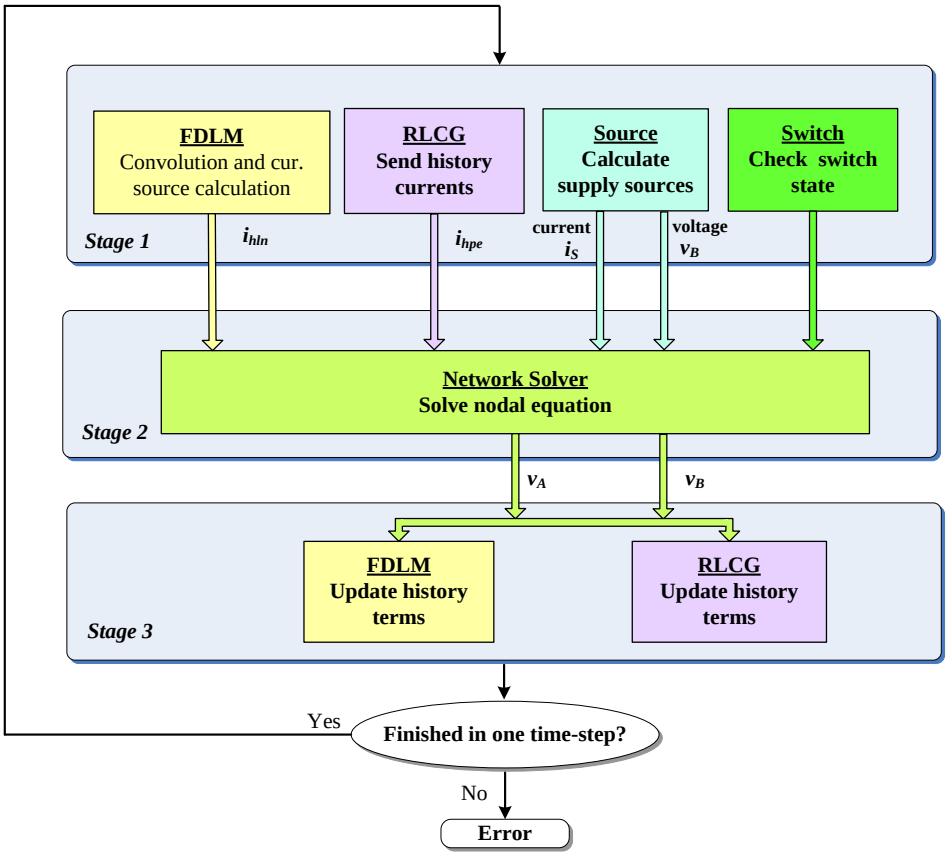


Figure 3.32: Parallelled real-time EMTP algorithm for FPGA implementation.

Stage 1 - Evaluating the nodal currents, *Stage 2* - solving the nodal equation, and *Stage 3* - updating the history terms. These three stages form an unavoidable sequentiality in the EMTP algorithm. For example, to solve the nodal equation, the nodal currents have to be known from *Stage 1*. Similarly, to update the history terms in *Stage 3*, we need the node voltages from *Stage 2*. Nevertheless, we can parallelize the operations within each of these stages. Fig. 3.32 shows the parallelled real-time EMTP algorithm flowchart. In *Stage 1* four operations are performed simultaneously: the calculation of the supply sources, calculation of transmission lines equivalent current sources at both sending and receiving ends, retrieving the RLCG elements' history current sources from memory, and checking switch states. Similarly, in *Stage 3* the history terms for the transmission lines and RLCG elements are updated simultaneously.

3.6.2 `MainControl` Module

The `MainControl` module, as shown in Fig. 3.33, coordinates the operation of the whole FPGA-based real-time EMT simulator to carry out the parallelled EMTP algorithm. It sends out control signals ($TLcmd$, $PEcmd$, $SOcmd$, $NWcmd$) to each module to perform

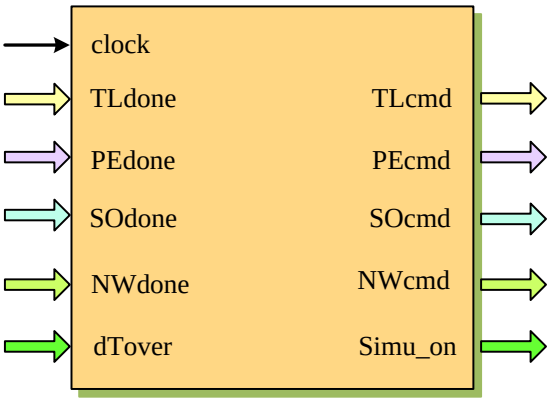


Figure 3.33: `MainControl` module and its input/output signals.

the required functions. Meanwhile, it receives the acknowledge signals ($TLdone$, $PEdone$, $SOdone$, $NWdone$) to judge if the function is done. Fig. 3.34 shows the finite state machine (FSM) diagram of the `MainControl` module.

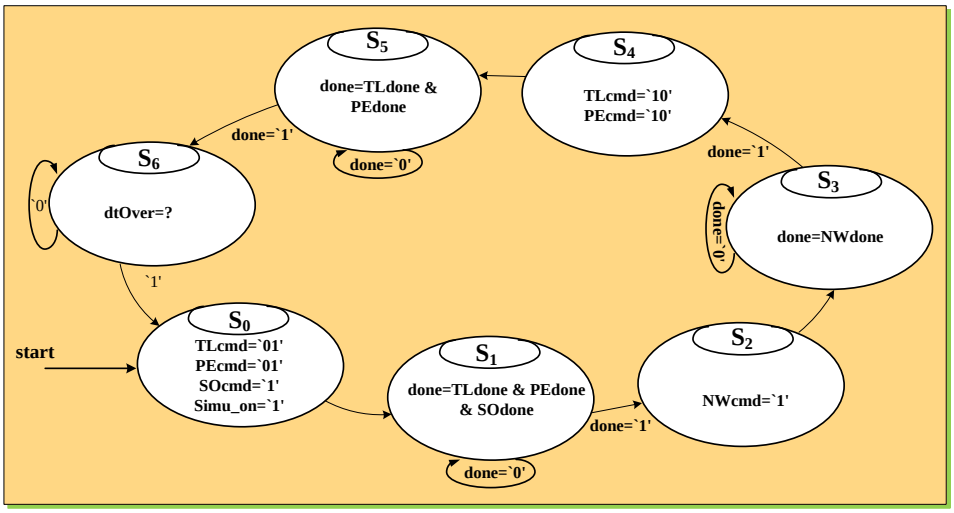


Figure 3.34: FSM diagram for parallelled real-time EMTP algorithm for FPGA implementation.

3.7 Implementation of Real-Time EMT Simulator on FPGA

The FPGA-based real-time EMT simulator is implemented on a Altera® Stratix™ S80 Development Board, shown in Fig. 3.35. The Stratix EP1S80 FPGA used on this board has the following features: 79,040 logic elements (LEs); 7,427,520 total RAM bits; 176 DSP blocks based on 9x9 multiplier; and 679 maximum user I/O pins.

Table 3.3 lists resources utilization by each module of the FPGA-based real-time EMT simulator. As can be seen from this table, the logic elements and DSP Blocks are almost

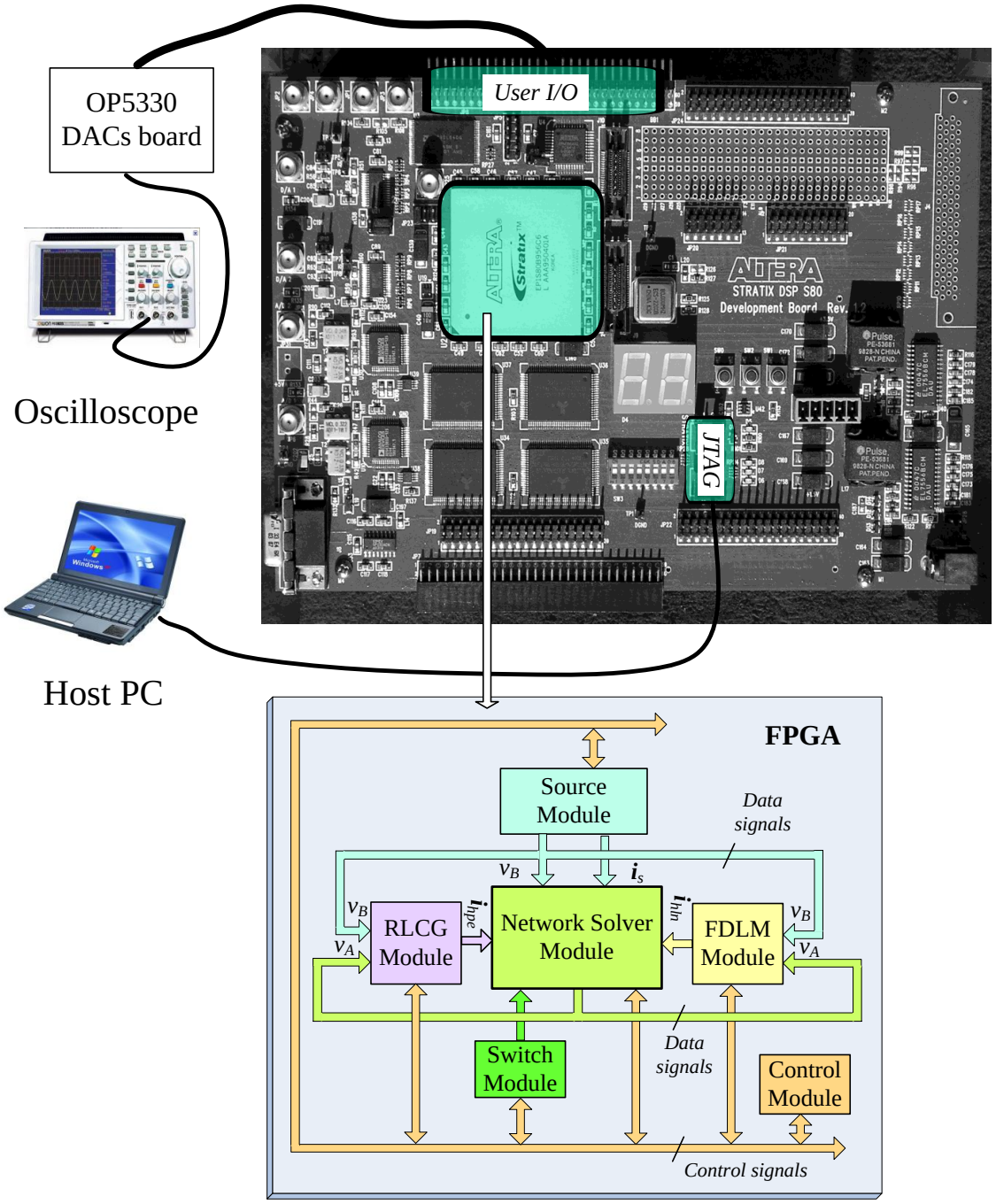


Figure 3.35: Real-time EMT simulator implemented on an Altera Stratix S80 FPGA development board.

fully utilized. Also as mentioned earlier, the `FDLM` module utilizes the most logic resources of the FPGA. The second most resource intensive module is the `Network Solver` module.

To simulate a power network in the proposed FPGA-based real-time EMT simulator,

Table 3.3: FPGA resources utilized by modules

Module	Logic Elements	DSP Blocks	Memory Bits
`FDLM`	37245 (47.1%)	96 (54.5%)	733632 (9.80%)
`RLCG`	10909 (13.8%)	40 (22.7%)	64896 (0.87%)
`Source`	2231 (2.8%)	16 (9.0%)	162278 (2.18%)
`Switch`	366 (0.4%)	0	768 (0.01%)
`Network Solver`	18588 (23.5%)	16 (9.0%)	49023 (1.04%)
`MainControl`	163 (0.2%)	0	0
Total	69502 (87.8%)	168 (95.2%)	1010597 (13.90%)

the network is first modeled in the off-line ATP software. The generated netlist is then analyzed and converted into various initial files of RAMs which are used to save the network parameters. Once the VHDL code for the simulator is compiled, the bitstream in the Host PC is downloaded into the FPGA development board through the JTAG interface. The simulation starts immediately once the download is finished. The real-time simulation results are sent to a DAC board connecting to the oscilloscope. The DAC board (OP5330 from OPAL-RT) has 16-bit resolution DACs and maximum $\pm$ 16V output voltage, so the real-time result in floating-point format has to be converted and truncated to a 16-bit binary before sending to the DAC. The truncation would cause loss of some information.

3.8 Real-Time EMT Simulation Case Study

An example of a power system is simulated to show the effectiveness of the proposed FPGA-based real-time EMT simulator. The system consists of 15 transmission lines, 4 generators, and 8 loads as shown in Fig. 3.36. The complete system data is listed in Appendix A. The transformers are modeled as the Thévenin equivalent networks. The lines are modeled using the frequency-dependent line model. Each of the system component is allocated to the appropriate FPGA module (Fig. 3.35) to process its calculation. For example, the 15 transmission line models are pipelined through the `FDLM` module. Fig. 3.37 shows the execution time for each stage of the paralleled EMTP algorithm (Fig. 3.32) for implementing this system. The total execution time is 11.213μs, while the actual time-step is 12μs. Based on the 12.5ns clock period used for the FPGA, this implies that it took 960 clock cycles to complete one loop of simulation for this case study. The achieved time-step Δt=12μs is at least 4 times smaller than the acceptable time-step of 50μs for transient simulation. As such it is possible to simulate a system that is at least 4 times as large as the present system, i.e., a system with 60 lines, 16 generators, and 32 loads can be simulated in real time with a time-step of Δt=50μs on this FPGA. We can also see from this figure, that the *Stage1* utilizes the most execution time. This is due to the convolution operation of FDLM as mentioned earlier. The time-step could be decreased significantly if more parallel `FDLM` modules are implemented on a larger FPGA.

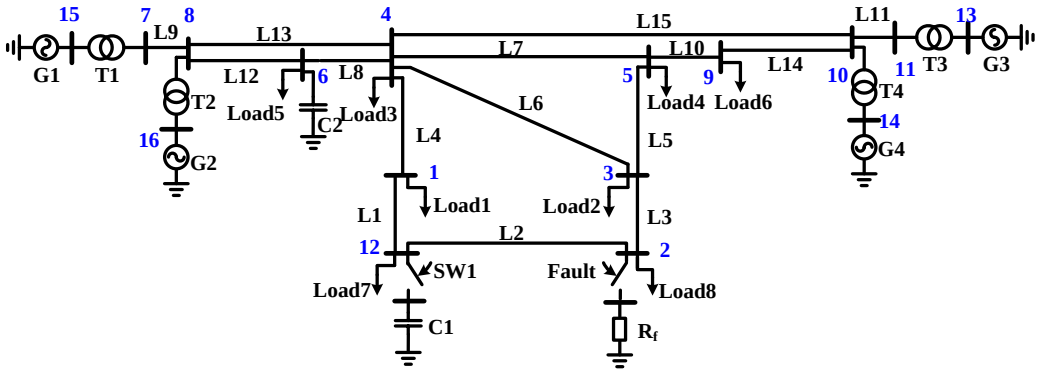


Figure 3.36: Single-line diagram of the power system used in the Case Study.

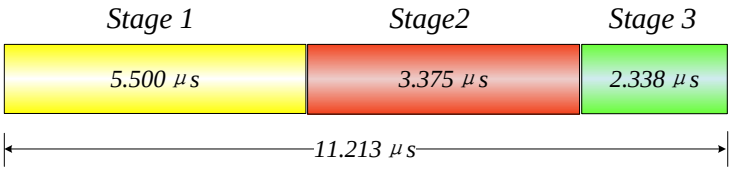


Figure 3.37: Execution time for each stage of the paralleled EMTP algorithm.

Two transient events are simulated. The first transient event is the capacitor $C1$ switched at Bus 12 at time $t = 0.05s$. Fig. 3.38 (a) and Fig. 3.38 (c) show the three-phase voltages and currents waveforms at Bus 12 captured by a real-time oscilloscope connected to the DACs. As can be seen in these figures, during the capacitor transient, voltage v_a drops from 110kV to almost 0kV immediately. A negative peak current of 1.5kA happens on i_b at $t = 0.05s$ and a positive peak current of 1.3kA appears on i_a at $t = 0.053s$. The transient lasts for about one cycle. Identical behavior can be observed from Fig. 3.38 (b) and Fig. 3.38 (d) which show the off-line ATP simulation results with a time-step of 12 μs. The second transient event is the Bus 2 three-phase-to-ground fault with 2Ω resistance to ground which occurs at $t = 0.05s$. Fig. 3.39 (a) and Fig. 3.39 (c) show the three-phase voltages and fault currents waveforms at Bus 12 captured by real-time oscilloscope. As can be seen, the voltages drop from 180kV peak to 60kV peak, and high frequency transient currents occur during the fault. The transient lasts for about 2 cycles. Again, detailed agreement between ATP off-line simulation shown in Fig. 3.39 (b) and Fig. 3.39 (d) and the FPGA-based real-time EMT simulator results can be observed.

3.9 Summary

In this chapter a FPGA-based real-time EMT simulator is described. The emulated power system components include linear lumped RLCG elements, transmission lines, supply

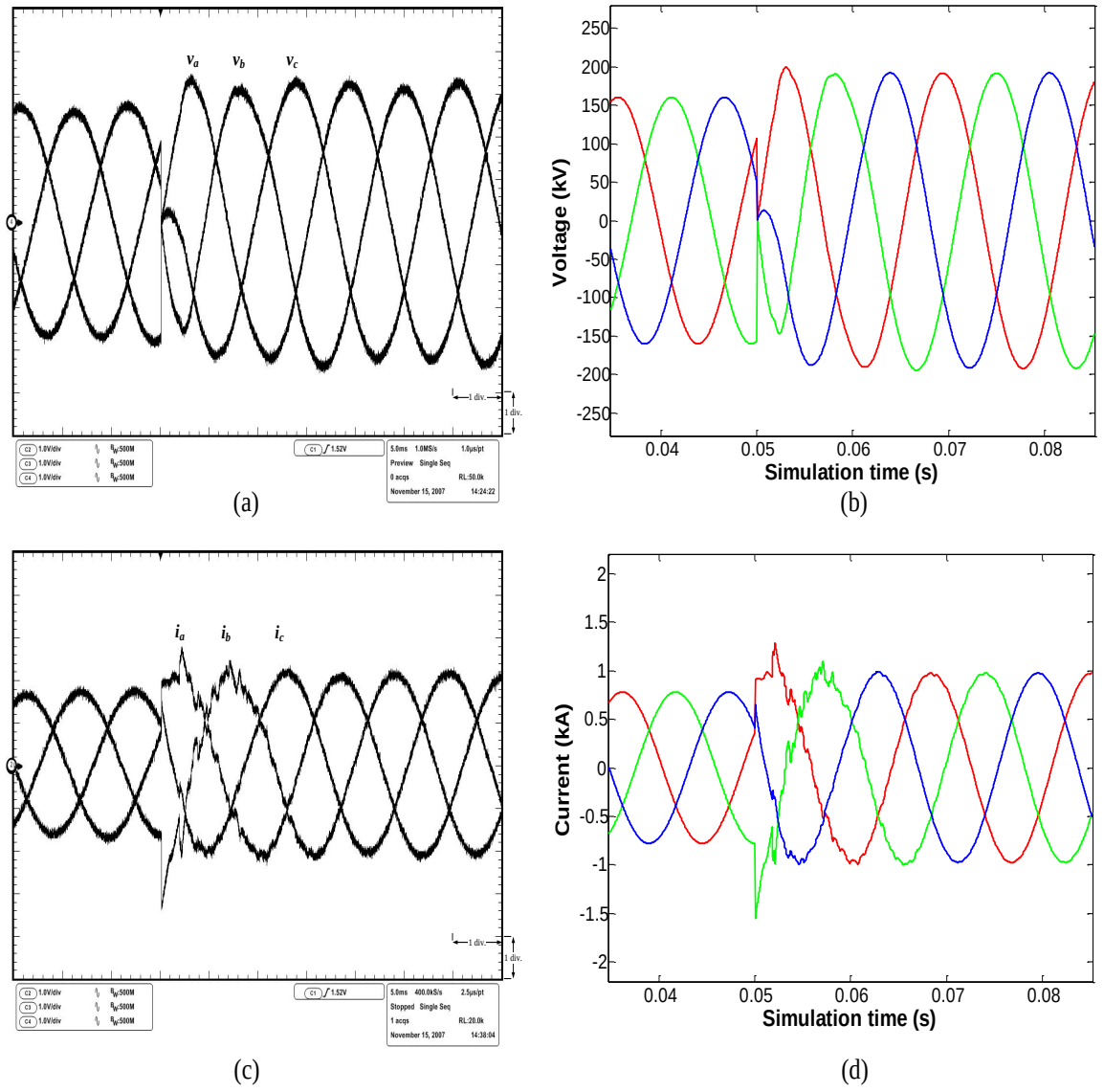


Figure 3.38: Real-time oscilloscope traces (a,c) and off-line simulation results from ATP (b,d) for a capacitor $C1$ switching transient at Bus 12. (a, b) Bus 12 voltages, (c, d) Bus 12 currents. Scale: x-axis: 1div. = 5ms, y-axis: (a) 1div. = 58kV, (c) 1div. = 0.44kA.

sources, circuit breakers. The central feature of this simulator is the frequency-dependent line model which allows for accurate line transient calculations. The network is solved efficiently on a proposed fast network solver hardware module exploiting sparse matrix technique. To fully exploit the parallel processing capacity of the FPGA, a paralleled EMTP algorithm is described which utilizes a deeply pipelined computation and a high precision floating-point number representation. An example of a power system with full frequency-dependent line modeling illustrates the accuracy of the FPGA-based real-time simulator. The transient results from the FPGA-based real-time simulator show excellent agreement with an off-line ATP simulation of the original system.

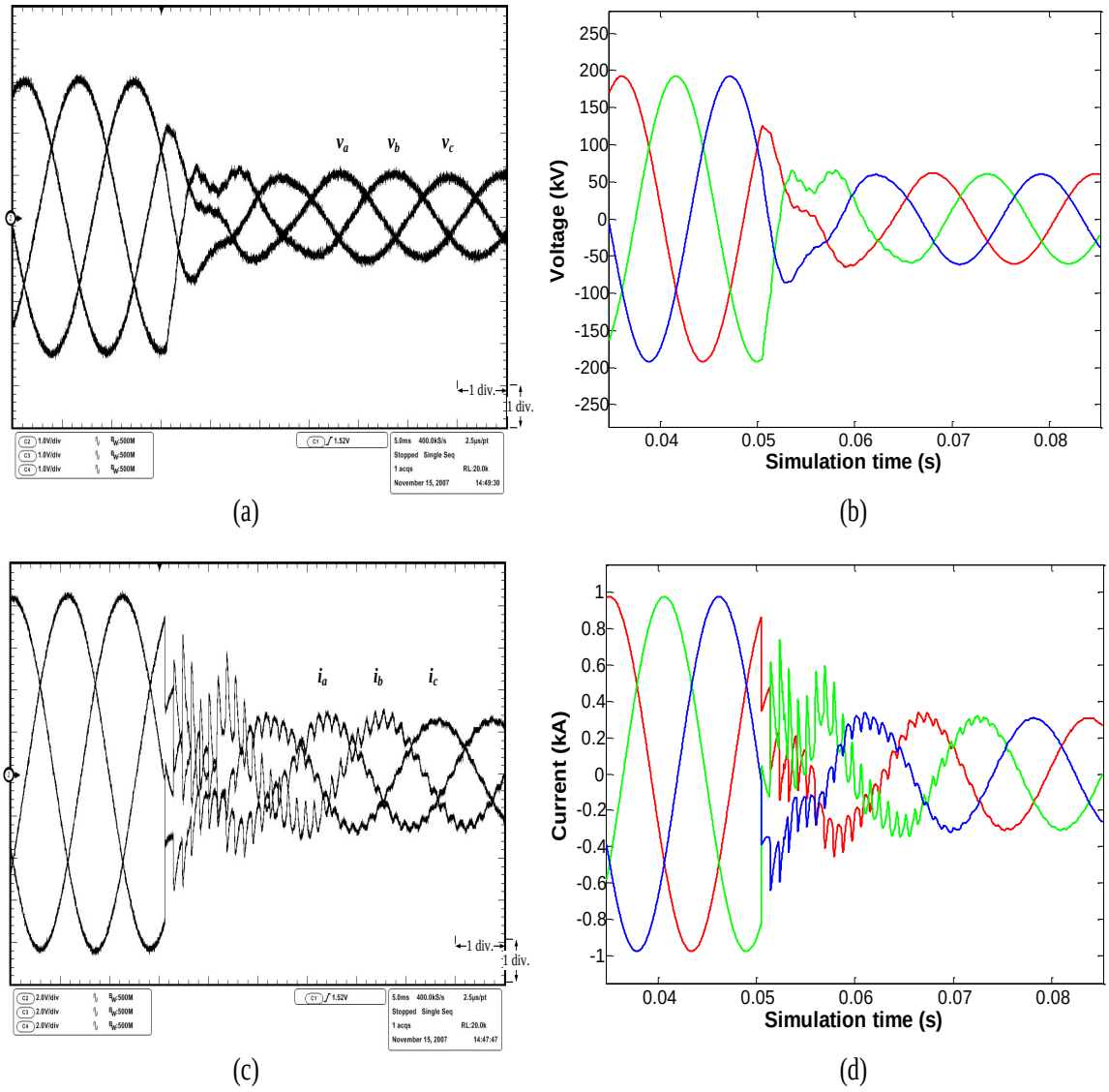


Figure 3.39: Real-time oscilloscope traces (a,c) and off-line simulation results from ATP (b,d) for a three-phase to ground fault transient at Bus 2. (a, b) Bus 12 voltages, (c, d) Bus 12 fault currents. Scale: x-axis: 1div. = 5ms, y-axis: (a) 1div. = 58kV, (c) 1div. = 0.22kA.

4

An Iterative Real-Time Nonlinear EMT Solver on FPGA

In this chapter[1] an iterative real-time nonlinear electromagnetic transient solver on FPGA is described to emulate the nonlinear elements in the power systems. Nonlinear elements play a significant role in the inception and propagation of transient overvoltages and overcurrents in electrical power systems. The commonly occurring nonlinear phenomena in power systems include magnetic saturation in transformers, ferroresonance, switching surges, and lightning strikes. Accurate simulation of nonlinear phenomena is vital from the perspective of such studies as insulation coordination, protection system design, power quality, and in general for maintaining the transmission and distribution infrastructure in a reliable working condition. In the off-line EMTP, the nonlinear solution has been well analyzed and implemented. However in real-time simulators, accurate simulation of nonlinear elements is very challenging due to the computational burden. The background on the nonlinear solution technique and on the Newton-Raphson algorithm is described first before giving the details of the hardware designs. Two real-time transient simulation case studies are presented. The real-time simulation results have been validated using off-line results from an ATP simulation.

4.1 Nonlinear Network Transient Solution

In off-line EMTP-type software programs such as ATP, EMTP-RV, and PSCAD/EMTDC, the representation of nonlinear elements is either through a piecewise linear approxima-

[1]Material from this chapter has been published: Y. Chen and V. Dinavahi, "An iterative real-time nonlinear electromagnetic transient solver on FPGA", *IEEE Trans. on Industrial Electronics*, vol. 58, no. 6, pp. 2547-2555, June 2011.

tion method (also known as pseudo nonlinear method) or an iterative method [4,5]. In the former the nonlinearity is represented by several linear segments. At each simulation time-step it is actually a specific linear element, such as a constant inductance, several of which can be used to represent, for example, the saturation characteristic of a transformer. However, the computational bottleneck occurs as the operating point switches from one linear segment to the next necessitating a recalculation of the system admittance matrix which results in a substantially longer run time for a large network. The frequent switching between linear segments is also known to cause numerical oscillations. For a more accurate simulation of nonlinear elements, an iterative method based on Newton-Raphson (N-R) can be used. Within each simulation time-step several N-R iterations are required for convergence, and each iteration entails a recalculation of the Jacobian matrix and solution of a set of linear equations. This process is very time-consuming. In an off-line simulator the computational burden can perhaps be overlooked, however, it becomes quite cumbersome in a real-time simulator. In a deterministic hard real-time system built using CPU or DSP based sequential hardware, an iterative nonlinear solution may not even be implementable using a requisite time-step mainly due to the uncertainty of convergence of the iterations within the time-step, and the limited computational power of such hardware. With FPGA being used for real-time EMT simulation, accurate real-time simulation of nonlinear element using iterative method is possible.

4.1.1 Compensation Method

With the inclusion of nonlinear elements, the power system network becomes a nonlinear system. In the EMTP, instead of solving the entire nonlinear network, the compensation method is commonly used to reduce computational burden. In compensation method [58], the nonlinear elements are first separated from the linear network, as shown in Fig. 4.1 (a), where p nonlinear elements connected to nodes ('ki') and ('mi') $\{i = 1, 2..., p\}$ have been extracted from a n-node linear network. Seen from these nodes the linear network can be represented as

$$\boldsymbol{v}_{km} = \boldsymbol{v}_{kmo} - \boldsymbol{R}_{eq}\boldsymbol{i}_{km}, \tag{4.1}$$

where $\boldsymbol{v}_{km}$ and $\boldsymbol{i}_{km}$ are vectors of p voltages across and currents through nonlinear elements, and $\boldsymbol{v}_{kmo}$ is a vector of p open-circuit voltages (without the nonlinear branches) between nodes ('ki') and ('mi'). $\boldsymbol{R}_{eq}$ is a $p \times p$ Thévenin equivalent resistance matrix of the linear network. To obtain $\boldsymbol{R}_{eq}$, vectors r_{thev_i} $(n \times 1)$ $\{i = 1, 2, ..., p\}$ which are the differences of the ki^{th} and mi^{th} columns of inverse admittance of the linear network $\boldsymbol{Y}^{-1}$ are first calculated. Then the (i, j) element of $\boldsymbol{R}_{eq}$ is given as

$$\boldsymbol{R}_{eq}(i, j) = [\boldsymbol{r}_{thev\_j}]_{ki} - [\boldsymbol{r}_{thev\_j}]_{mi}. \tag{4.2}$$

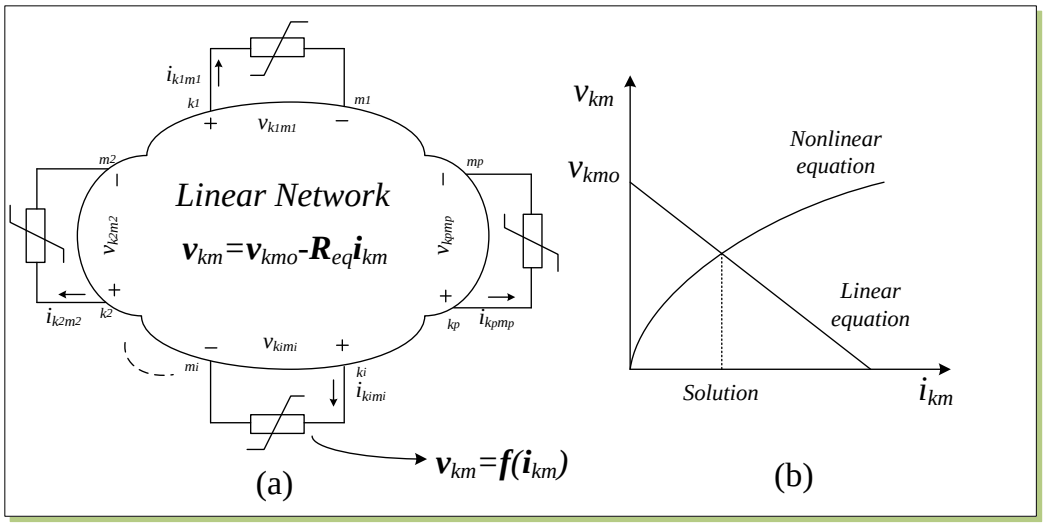


Figure 4.1: (a) Network with p nonlinear elements, and (b) illustration of compensation method.

The $\boldsymbol{v}$ and $\boldsymbol{i}$ characteristics of the nonlinear elements are expressed as

$$\boldsymbol{v}_{km} = \boldsymbol{f}(\boldsymbol{i}_{km}), \tag{4.3}$$

where $\boldsymbol{f}$ is the nonlinear functions.

When (4.1) and (4.3) are solved simultaneously as illustrated graphically in Fig. 4.1 (b), the current $\boldsymbol{i}_{km}$ has been obtained. Then $\boldsymbol{i}_{km}$ is superimposed on the linear network as a current source, thus the entire network is solved as follows

$$\boldsymbol{v}_c = \boldsymbol{v}_o - \boldsymbol{R}_{thev}\boldsymbol{i}_{km}, \tag{4.4}$$

where $\boldsymbol{v}_c$ is the compensated final node voltages, and $\boldsymbol{v}_o$ is the node voltages without nonlinear elements. $\boldsymbol{R}_{thev}$ is a $n \times p$ resistance matrix expressed as

$$\boldsymbol{R}_{thev} = [\ \boldsymbol{r}_{thev\_1} \quad \boldsymbol{r}_{thev\_2} \quad \ldots \quad \boldsymbol{r}_{thev\_p}\ ]. \tag{4.5}$$

The merits of compensation method is to solve a small size of nonlinear equations. The size is determined by the number of nonlinear elements p. In the three-phase power system if there are M three-phase nonlinear elements, the total number of nonlinear equations is $p = 3 \times M$. Fortunately, the three-phase nonlinear elements are commonly separated by transmission lines. In such case, due to the traveling time delay of transmission lines the M three-phase nonlinear elements are decoupled with each other, which implies that M sets of 3 nonlinear functions can be solved simultaneously in a parallel computational environment.

4.1.2 Newton-Raphson Method

The Newton-Raphson method is widely used to solve the nonlinear equations due to its quadratic convergence. Substituting (4.3) into (4.1) the nonlinear equations are obtained as

$$\boldsymbol{F}(\boldsymbol{i}_{km}) \equiv \boldsymbol{f}(\boldsymbol{i}_{km}) - \boldsymbol{v}_{kmo} + \boldsymbol{R}_{eq}\boldsymbol{i}_{km} = \boldsymbol{0}. \tag{4.6}$$

The objective of Newton-Raphson method is to find solution $\boldsymbol{i}_{km}$. By applying the first order Taylor series expansion for the nonlinear equations (4.6), the updated solution is obtained by solving the following system of linear equations

$$\boldsymbol{J}(\boldsymbol{i}_{km}^{k+1} - \boldsymbol{i}_{km}^{k}) = -\boldsymbol{F}(\boldsymbol{i}_{km}^{k}), \tag{4.7}$$

where $\boldsymbol{J}$ is Jacobian matrix, $\boldsymbol{i}_{km}^{k+1}$ and $\boldsymbol{i}_{km}^{k}$ are the current vectors at the $(k+1)^{th}$ and k^{th} iterations, respectively. This method is referred to as the continuous Newton-Raphson (CNR) method in this design since the nonlinear function is a continuous analytical equation. The Jacobian matrix and nonlinear equations are computed at each iteration using

$$\begin{cases} \boldsymbol{J} = \frac{\partial \boldsymbol{F}(\boldsymbol{i}_{km})}{\partial \boldsymbol{i}_{km}} = \boldsymbol{R}_{eq} + \frac{\partial \boldsymbol{f}(\boldsymbol{i}_{km})}{\partial \boldsymbol{i}_{km}}, \\ -\boldsymbol{F}(\boldsymbol{i}_{km}^{k}) = \boldsymbol{v}_{kmo} - \boldsymbol{R}_{eq}\boldsymbol{i}_{km}^{k} - \boldsymbol{f}(\boldsymbol{i}_{km}^{k}). \end{cases} \tag{4.8}$$

The convergence criteria is defined as

$$\|\boldsymbol{i}_{km}^{k+1} - \boldsymbol{i}_{km}^{k}\| < \varepsilon_1 \; and \; \|\boldsymbol{F}(\boldsymbol{i}_{km}^{k+1})\| < \varepsilon_2, \tag{4.9}$$

with ε_1 and ε_2 set to sufficiently small values.

If the nonlinear function is given by a piecewise linear curve, the method becomes piecewise Newton-Raphson (PNR) [59]. As shown in Fig. 4.2 (a), each linear segment of the piecewise curve can be defined by

$$\boldsymbol{v}_{km} = \boldsymbol{e}_{kmj} + \boldsymbol{R}_j\boldsymbol{i}_{km}, \boldsymbol{i}_{km} \in [\boldsymbol{I}_{kmj-}, \boldsymbol{I}_{kmj+}], \tag{4.10}$$

where $\boldsymbol{e}_{kmj}$ and $\boldsymbol{R}_j$ are the intercept voltage and resistance of the j^{th} segment of the curve, which falls within the interval $\boldsymbol{I}_{kmj-}$ and $\boldsymbol{I}_{kmj+}$.

Substituting (4.10) into (4.1) the nonlinear equations become

$$\boldsymbol{F}(\boldsymbol{i}_{km}) \equiv (\boldsymbol{v}_{kmo} - \boldsymbol{R}_{eq}\boldsymbol{i}_{km}) - (\boldsymbol{e}_{kmj} + \boldsymbol{R}_j\boldsymbol{i}_{km}) = \boldsymbol{0}. \tag{4.11}$$

Using (4.7) and (4.8) directly, we obtain

$$\begin{aligned} (-\boldsymbol{R}_{eq} - \boldsymbol{R}_j)(\boldsymbol{i}_{km}^{k+1} - \boldsymbol{i}_{km}^{k}) &= (\boldsymbol{e}_{kmj} + \boldsymbol{R}_j\boldsymbol{i}_{km}^{k}) - (\boldsymbol{v}_{kmo} - \boldsymbol{R}_{eq}\boldsymbol{i}_{km}^{k}), \\ (\boldsymbol{R}_{eq} + \boldsymbol{R}_j)\boldsymbol{i}_{km}^{k+1} &= \boldsymbol{v}_{kmo} - \boldsymbol{e}_{kmj}. \end{aligned} \tag{4.12}$$

It is important to observe that the term $\boldsymbol{i}_{km}^{k}$ is canceled out in (4.12), which means that the calculation of Jacobian matrix $\boldsymbol{J}$ is not required in the PNR method. This is because $\boldsymbol{R}_j$ which is the derivatives of the piecewise linear curve have been precalculated. Thus the

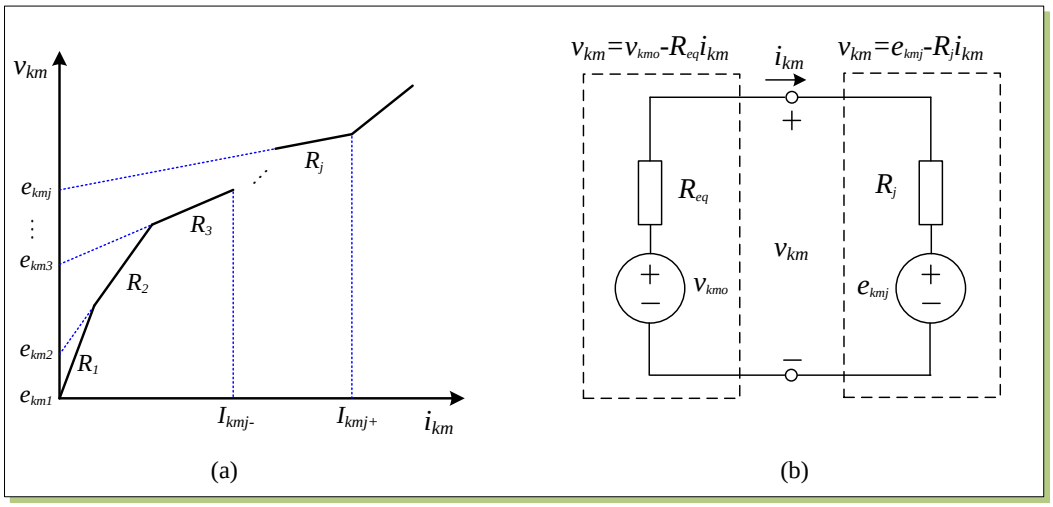


Figure 4.2: (a) Piecewise linear function, and (b) its implementation in PNR.

computational burden of PNR is much less than that of CNR at cost of reduced accuracy. Since (4.10) is for the j^{th} segment of the piece-wise curve, the solution is then checked to see if the solution satisfies $[\boldsymbol{I}_{kmj-}, \boldsymbol{I}_{kmj+}]$. If it does, a valid solution has been found; otherwise, the next iteration starts.

If the nonlinear element is a nonlinear inductance, its nonlinear flux-current characteristic is given as

$$\boldsymbol{\lambda} = \boldsymbol{f}(\boldsymbol{i}_{km}). \tag{4.13}$$

Applying the Trapezoidal rule of integration to (4.13) yields

$$\boldsymbol{\lambda}(t) = \int \boldsymbol{v}_{km}(t)dt, \tag{4.14}$$

the linear flux-voltage characteristic is obtained as

$$\boldsymbol{\lambda} = \frac{\Delta t}{2}\boldsymbol{v}_{km}(t) + \boldsymbol{\lambda}_{his}(t - \Delta t), \tag{4.15}$$

where the history term is given as

$$\boldsymbol{\lambda}_{his}(t - \Delta t) = \boldsymbol{\lambda}(t - \Delta t) + \frac{\Delta t}{2}\boldsymbol{v}_{km}(t - \Delta t). \tag{4.16}$$

Then the voltage-current characteristic of nonlinear inductance is obtained by substituting (4.15) into (4.13).

4.2 Real-Time Hardware Emulation of Nonlinear Solver on FPGA

A hardware module `NR` is designed to emulate nonlinear elements using iterative Newton-Raphson method in the FPGA. The symbol of this module and its iuput/output signals is shown in Fig. 4.3. The signal `Vn` receives open circuit voltages $\boldsymbol{v}_0$ form `Network Solver`

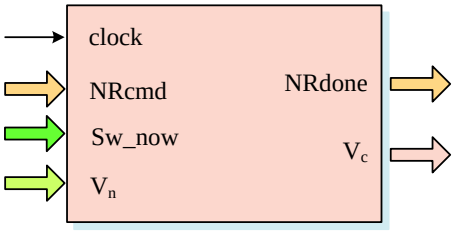


Figure 4.3: `NR` module and its input/output signals.

module (Section 3.5.2). The calculated compensated voltage $\boldsymbol{v}_c$ is sent through signal `Vc`. `NRcmd` and `NRdone` are control and acknowledge signals. The details of the architecture and functions of this module are described as follows.

4.2.1 Hardware Architecture and Parallelism

As shown in Fig. 4.4, the hardware architecture of `NR` module consists of four main hardware submodules: (1) `NLFunc` (evaluating nonlinear function $\boldsymbol{f}(\boldsymbol{i}_{km})$ and $\partial \boldsymbol{f}(\boldsymbol{i}_{km})/\partial \boldsymbol{i}_{km}$; (2) `CompJF` (computing $\boldsymbol{J}$ and $-\boldsymbol{F}(\boldsymbol{i}_{km})$ (4.8)); (3) `GJE` (Gauss-Jordan Elimination) for solving (4.7) or (4.12) for $\boldsymbol{i}_{km}$; and (4) `CompVc` (computing the compensated node voltage $\boldsymbol{v}_c$ (4.4)).

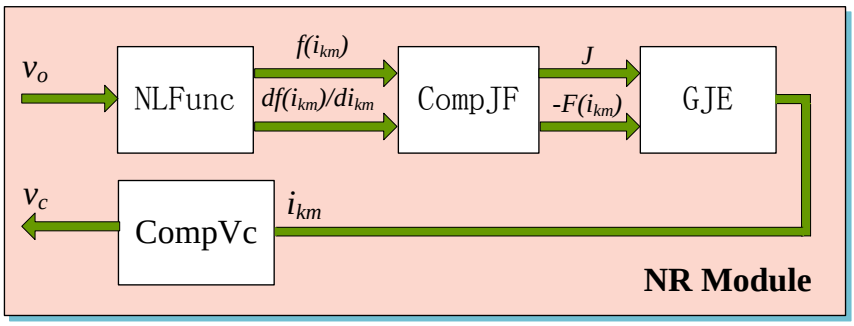


Figure 4.4: Overall architecture of the nonlinear solver in the FPGA.

The overall algorithm of N-R solution is in sequential; that is, one N-R iteration starts with calculating $\boldsymbol{f}(\boldsymbol{i}_{km})$ and $\partial \boldsymbol{f}(\boldsymbol{i}_{km})/\partial \boldsymbol{i}_{km}$; then the $\boldsymbol{J}$ and $-\boldsymbol{F}(\boldsymbol{i}_{km})$ can be calculated; After that the $\boldsymbol{i}_{km}$ is solved using Gauss-Jordan elimination method. Finally the compensated node voltages $\boldsymbol{v}_c$ is calculated. The convergence check determines if the iteration is finished or not using (4.9). However, the parallel operations still exist in each step of iteration. Fig. 4.5 shows the finite state machine (FSM) diagram for `NR` module. The sequential operations are clearly shown by state transitions $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_0$. The possible parallel operations, which take advantage of the hardware parallelism of the FPGA, are also shown in Fig. 4.5. For example, in state S_0, the nonlinear function evaluations $\boldsymbol{f}(\boldsymbol{i}_{km})$ and $\partial \boldsymbol{f}(\boldsymbol{i}_{km})/\partial \boldsymbol{i}_{km}$ are performed simultaneously. The computation of $\boldsymbol{J}$ and

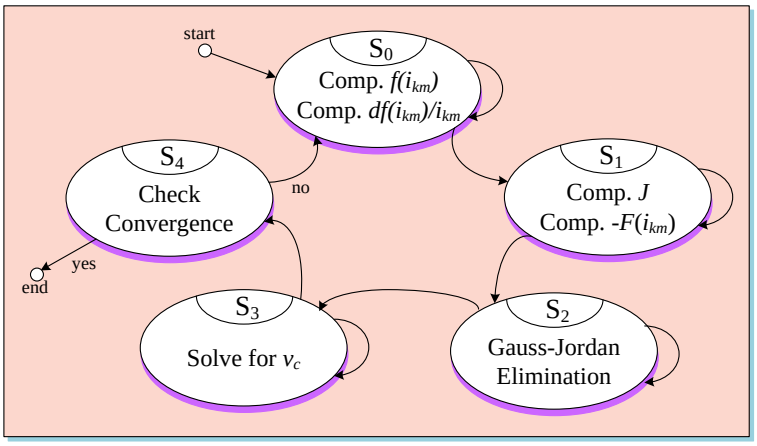


Figure 4.5: Finite state machine (FSM) diagram of `NR` module.

$-\boldsymbol{F}(\boldsymbol{i}_{km})$ in (4.8) are also processed concurrently in state S_1. Meanwhile, the Gauss-Jordan elimination procedure is paralleled in state S_2 as discussed in more detail later.

4.2.2 Floating-Point Nonlinear Function Evaluation

At each N-R iteration, nonlinear function evaluation is required. The computational burden of this evaluation depends on the number of nonlinear elements in the original system and the nature of these nonlinearities. The commonly used methods for nonlinear function evaluation on the FPGA include the look-up table (LUT) method and other approximation methods such as COordinate Rotation DIgital Computer (CORDIC), series expansion, and the regular N-R iteration (an inner iterative loop inside the outer iteration) [60, 61]. Many of them have been implemented in hardware, however, based on fixed-point format mostly. The floating-point implementation is still cumbersome due to its large latency and logic resource utilization. Among the above methods, the LUT is still widely used due to its speed and convenience of implementation. To reduce the size of LUT and improve the accuracy, linear interpolation has been used in this design to compute intermediate values between two locations of the LUT. Fig. 4.6 shows the hardware design of the `NLFunc` submodule. It consists of an `Address generation` unit, a dual-port RAM unit serving as a `LUT`, and a `Linear interpolation` unit. The `Address generation` unit has been utilized in `Source` module in Section 3.3.2, it is not discussed here again.

After $f(x_i)$ and $f(x_{i+1})$ are retrieved from the LUT the linear interpolation is employed to calculate $f(x)$. The hardware design of `Linear interpolation` unit is shown in Fig. 4.7 (b). It utilizes 2 floating-point subtractors ($y = a - b$) and multiplier-adder ($y = a \times b + c \times d$). The calculation is quite straightforward as illustrated in Fig. 4.7 (a).

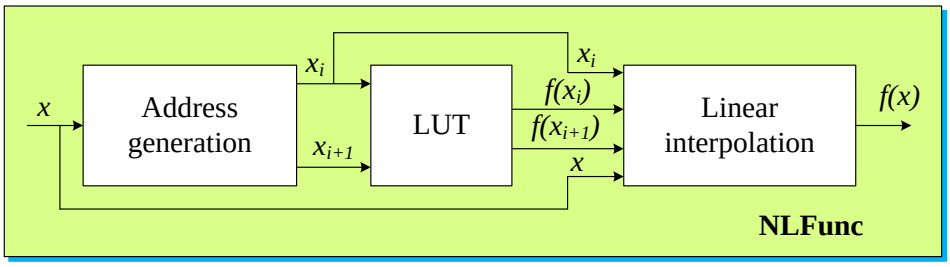
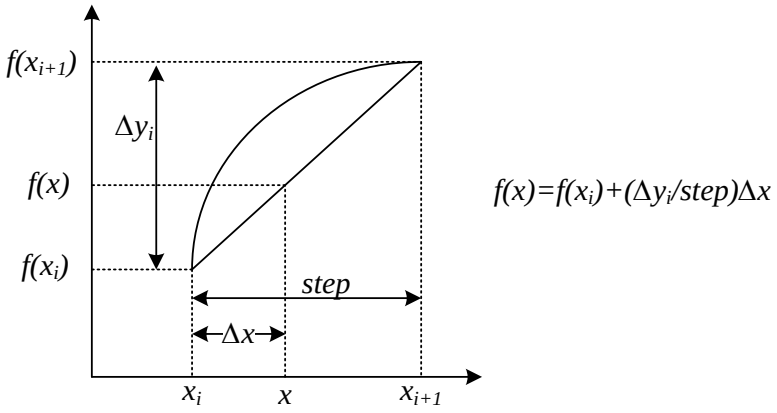
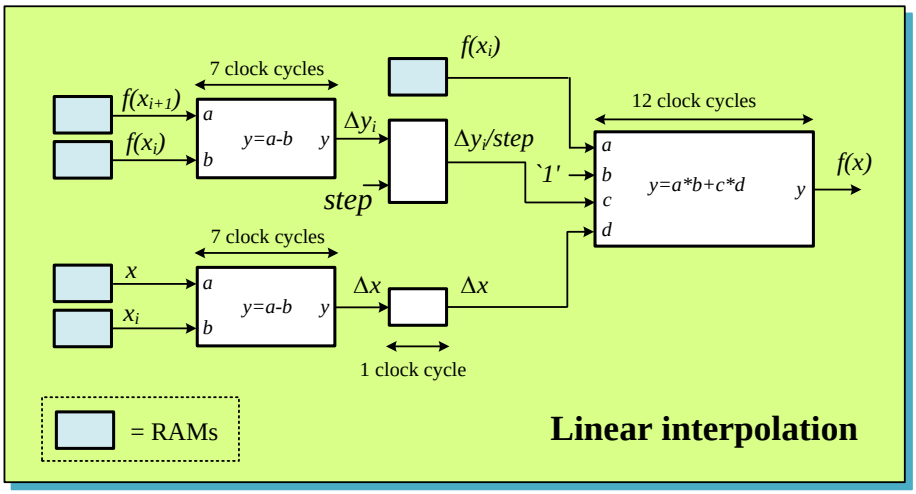


Figure 4.6: Floating-point nonlinear function computation using LUT and linear interpolation.



(a)



(b)

Figure 4.7: (a) Linear interpolation of $f(x)$, and (b) its pipelined computation scheme.

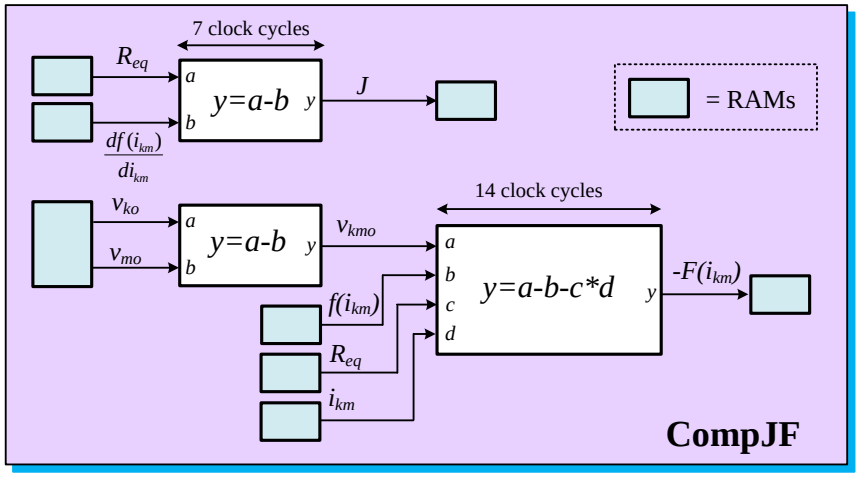


Figure 4.8: Parallel computational scheme for calculating $\boldsymbol{J}$ and $-\boldsymbol{F}(\boldsymbol{i}_{km})$.

4.2.3 Computing $\boldsymbol{J}$ and $-\boldsymbol{F}(\boldsymbol{i}_{km})$ in Parallel

Since $\boldsymbol{f}(\boldsymbol{i}_{km})$ and $\partial \boldsymbol{f}(\boldsymbol{i}_{km})/\partial \boldsymbol{i}_{km}$ being calculated, the computations of $\boldsymbol{J}$ and $-\boldsymbol{F}(\boldsymbol{i}_{km})$ (4.8) are carried out in parallel as shown in Fig. 4.8. The computations are quite straightforward. $\boldsymbol{J}$ is calculated by an adder. $-\boldsymbol{F}(\boldsymbol{i}_{km})$ is calculated by an arithmetic unit ($y = a - b - c \times d$) after $\boldsymbol{v}_{km0}$ being computed by a subtractor.

4.2.4 Parallel Gauss-Jordan Elimination

The set of linear algebraic equations in (4.7) or (4.12) are solved by Gauss-Jordan elimination (GJE). Compared to other elimination methods such as Gaussian elimination with backward substitution and LU decomposition, the GJE has more number of operations than those of other two methods; however, the size of linear algebraic equations is only 3 under the situation that the three-phase nonlinear elements are separated by transmission lines as stated early. This makes the number of operation almost no difference in three linear solution methods. The Gauss-Jordan elimination is employed in this design because the algorithm is straightforward, which makes it easier for hardware implementation.

To simulate M three-phase nonlinear elements, M sets of 3 linear equations need to be solved within each iteration of each simulation time-step. For a large M, the sequential GJE can be very time consuming. Hence, a parallelled and pipelined scheme has been designed to speed up this process, as shown in Fig. 4.9. It consists of 3 elimination units (`Eli1`, `Eli2`, and `Eli3`) and a factorization unit (`Fac`). Before the GJE process begins, the 3×3 matrix $\boldsymbol{J}$ and 3×1 vector $-\boldsymbol{F}(\boldsymbol{i}_{km})$ for each three-phase nonlinear element are augmented together as a 3×4 matrix. For M three-phase nonlinear elements, M sets of 3×4 matrices are combined together by rows to get a $3 \times 4M$ matrix whose each row is stored in the RAM of 3 elimination units. Then the GJE computation proceeds according to the following two steps:

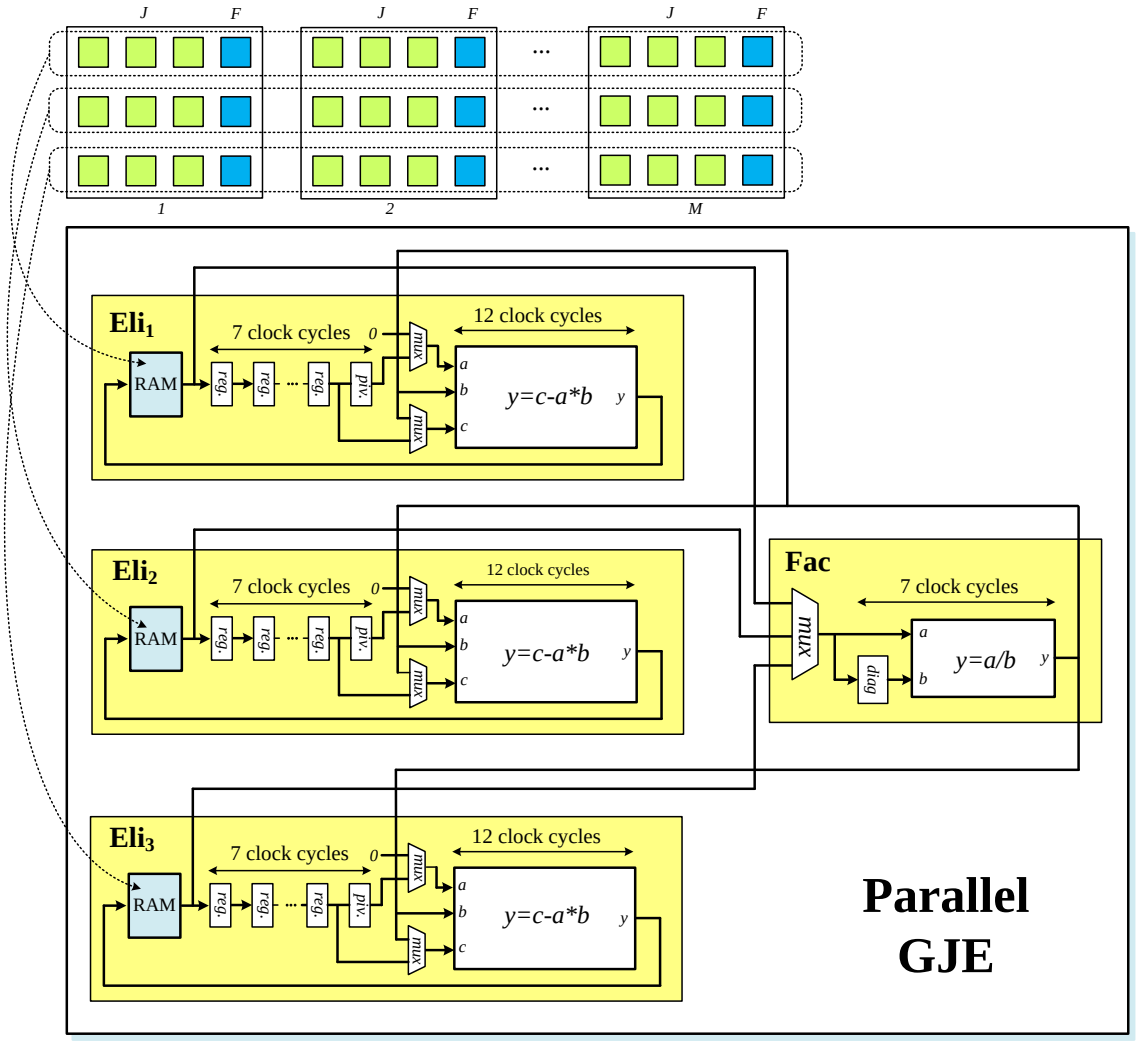


Figure 4.9: Hardware design of Parallel Gauss-Jordan elimination.

1. Factorization: The i^{th} ($i = 1, 2, 3$) row is retrieved from the corresponding elimination unit, and the diagonal element is identified and registered as *diag* in the `Fac` unit. Then the remaining elements within the row are divided by the registered *diag*, and the factorized row is sent back to all elimination units.

2. Elimination: The elimination is done simultaneously in all elimination units. In the i^{th} elimination unit elimination is not needed, so the factorized row is just saved in its RAM. In other two elimination units the i^{th} element of corresponding row is recognized and registered as *piv*, then multiplied by the elements of factorized row. The resulting product is then added to the corresponding element of row in its RAM.

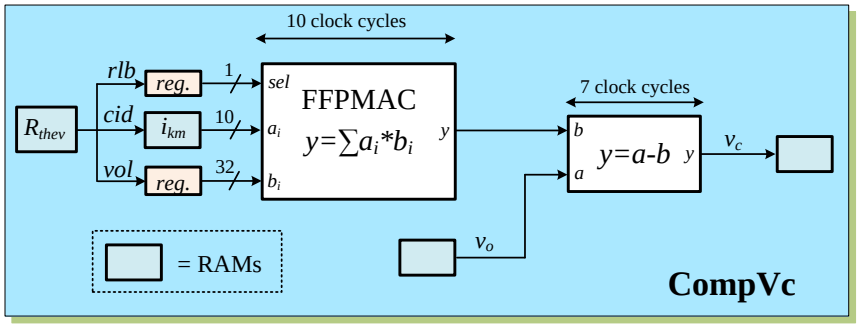


Figure 4.10: Pipelined computational scheme for calculating $\boldsymbol{v}_C$.

4.2.5 Computing $\boldsymbol{v}_c$

Finally the complete node voltages can be calculated. A hardware submodule is designed to calculate $\boldsymbol{v}_c$, as shown in Fig. 4.10. Since $\boldsymbol{R}_{thev}$ ia a sparse matrix, sparse matrix technique is employed here. This is realized by a `FFPMAC` unit along with a sparse matrix storage format scheme which has been discussed in detail in `Network Solver` module in Section 3.5.2.

4.3 FPGA-Based Nonlinear Transient Simulation

4.3.1 FPGA Hardware Implementation

The iterative real-time nonlinear electromagnetic transient solver is implemented on a Altera Stratix III Development Board DE3, as shown in Fig. 4.11. The FPGA used on this board is a Stratix III EP3SL150 which has the following main features: 142,500 equivalent logic elements (LEs); 6,390 total RAM Kbits; 384 18x18-bit multiplier; 8 phase-locked loop (PLL), and 1152 maximum user I/O pins. A DAC card is connected to the DE3 board. The DAC has 14-bit resolution, 125 MSPS data rate, and $\pm$ 500mV output voltage range. The simulation results in 32-bit floating-point format are truncated to 14-bit binary before sending to DAC, causing the loss of some information..

4.3.2 Case Studies

Two case studies are used to show the effectiveness of the proposed iterative real-time nonlinear transient solver on the FPGA. The first case study is a three-phase series-compensated transmission system whose single line diagram is shown in Fig. 4.12. The compensation capacitor C connected in series between two lines ($line1$ and $line2$) is protected by a surge arrester. The lines are modeled using distributed parameter line model (Bergeron's model) to capture the traveling waves due to transients on the system. The surge arresters are

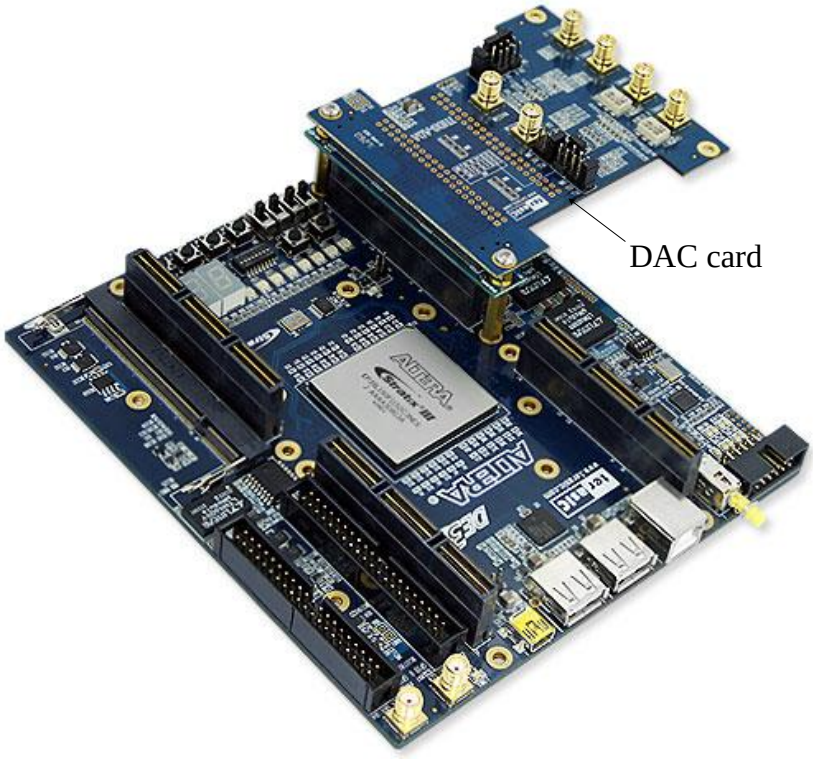


Figure 4.11: Altera Stratix III development board DE3 and connected DAC card.

highly nonlinear resistors characterized by

$$i = p(\frac{v}{V_{ref}})^q, \tag{4.17}$$

where q is the exponent, V_{ref} and p are arbitrary reference values. The elements R_{L1}, L_{L1}, and R_{L2} represent a composite load, and R_f is the fault resistance. The CNR method is used to solve the nonlinear equations. The complete system data is listed in Appendix B.1.

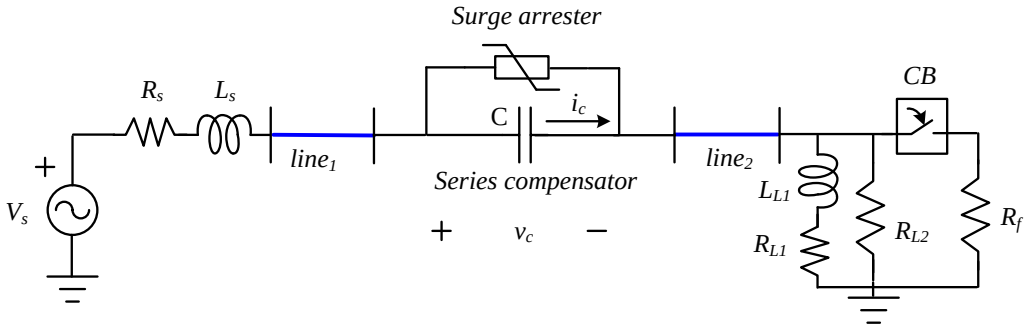
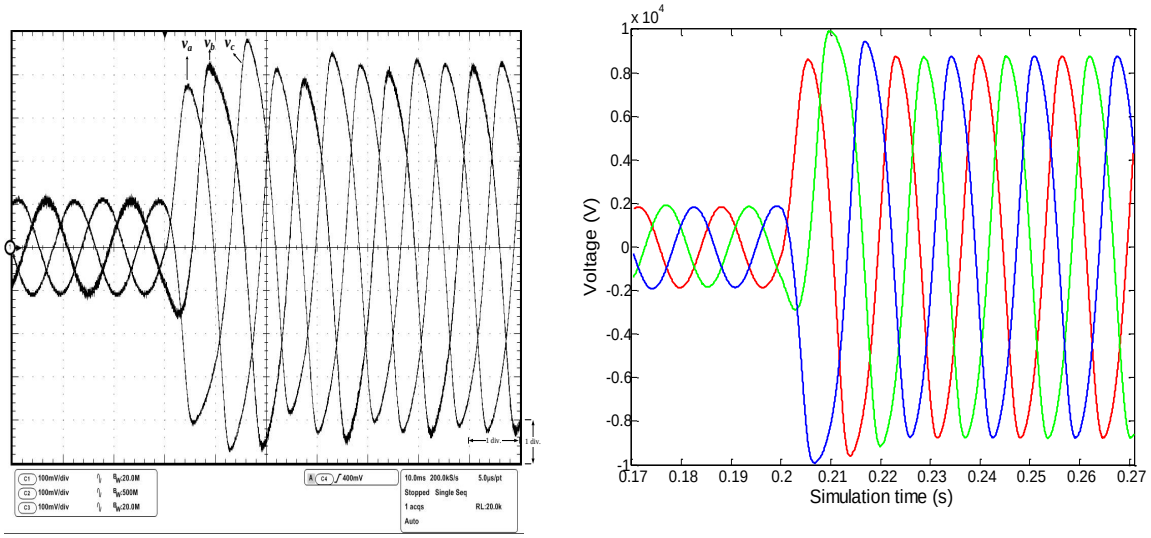


Figure 4.12: Single-line diagram for Case Study I (Surge arrester transient in a series compensated transmission system).

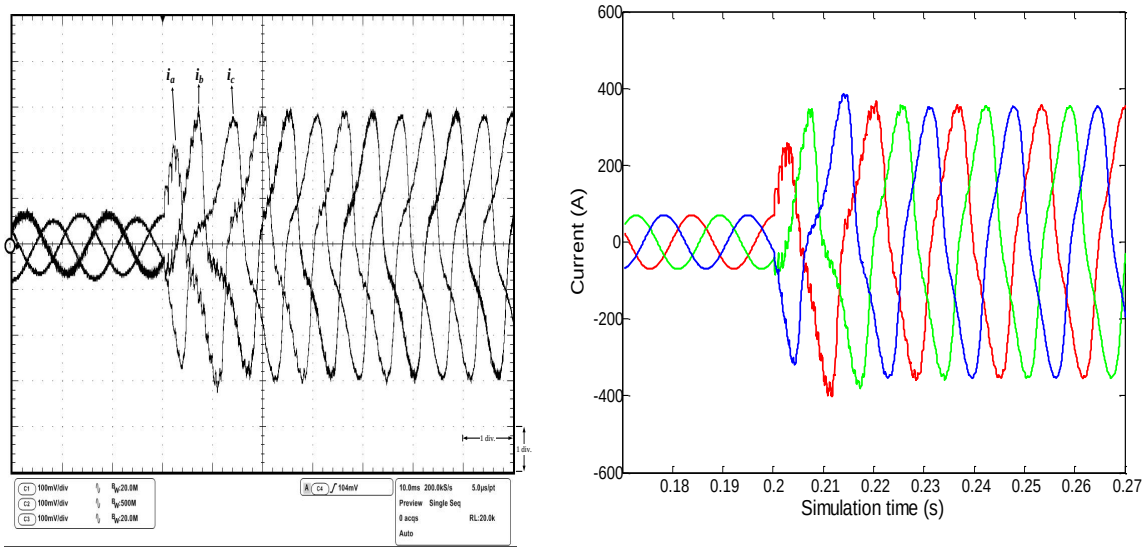


(a) (b)

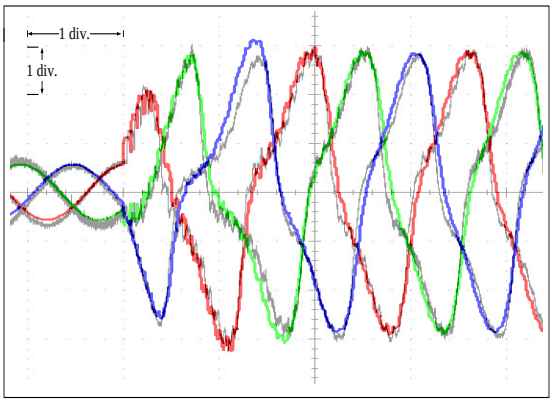
Figure 4.13: Real-time oscilloscope traces (a) and off-line simulation results from ATP (b) of the three-phase voltages across the surge arresters for a three-phase fault. Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 2kV.

A three-phase fault is applied at the load terminals at t=0.2 seconds by closing the circuit breaker CB. The current increases in the series capacitor and produces an overvoltage that is limited by the surge arresters. Fig. 4.13 (a) shows the three-phase voltage transients across the surge arresters captured from a real-time oscilloscope connected to the DAC card on the Stratix III FPGA board. The peak overvoltage is limited to 8kV compared to 15kV without the arrester. Fig. 4.14 (a) shows the three-phase transient currents in the surge arresters in the real-time simulation. Large currents drawn by the arresters can be observed. Similar behavior can be observed from Fig. 4.13 (b) and Fig. 4.14 (b) which show the off-line ATP simulation results. The difference between the off-line and real-time results might be caused by the different initial values and switching time.

The second case study illustrates ferroresonance in the three-phase voltage transformers. Fig. 4.15 (a) and (b) show the single-line and equivalent network diagrams of this case study, respectively. The voltage transformer is connected to the power system represented by a voltage source V_s through a circuit breaker CB. C_w is the circuit breaker's grading capacitance, C_s is the total phase-to-earth capacitance including transformer winding capacitance. The resister R_{fe} represents transformer core losses. The transformer current is represented by its piecewise nonlinear magnetization characteristic shown in Fig. 4.16; thus the PNR is used in this case study with 5 linear segments to represent the nonlinearity. The complete system data is also listed in Appendix B.2. The ferroresonance response is verified by the opening the circuit breaker caused by a three-phase fault on the transformer at t=0.2 seconds. Again similar behavior of real-time and off-line simulations can be observed from Fig. 4.17.



(a) (b)



(c)

Figure 4.14: Real-time oscilloscope traces (a), off-line simulation results from ATP (b), and zoomed and superimposed view (c) of the three-phase currents in the surge arresters for a three-phase fault. Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 128A.

Fig. 4.18 (a) and (b) show the execution time for each state in the finite state machine in Fig. 4.5 for the two case studies. The execution time for "others" include time used for calculating transmission lines, $RLCG$ elements, and sources, etc. Based on the 60MHz FPGA input clock the total execution time for Case Study I is 4.91μs for the average of 3 iterations, while the actual time-step is 5μs. For Case Study II, since there are no calculations for the Jacobin matrix the execution time is only 2.89μs, while the actual time-step is 3μs.

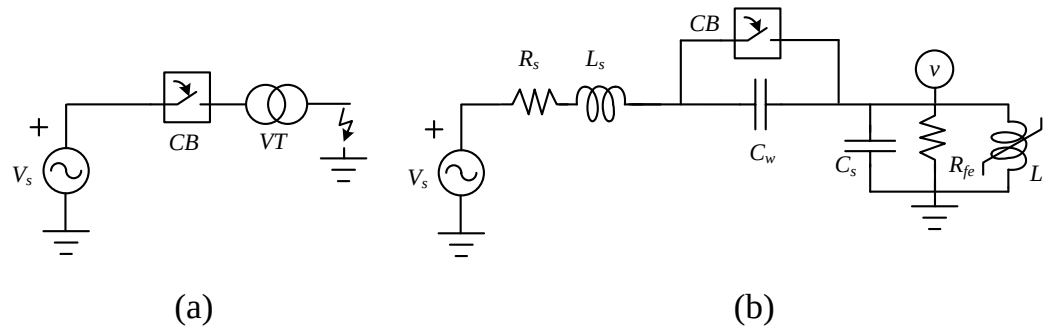


Figure 4.15: (a) Single-line diagram, and (b) equivalent network diagram for Case Study II (ferroresonance transient).

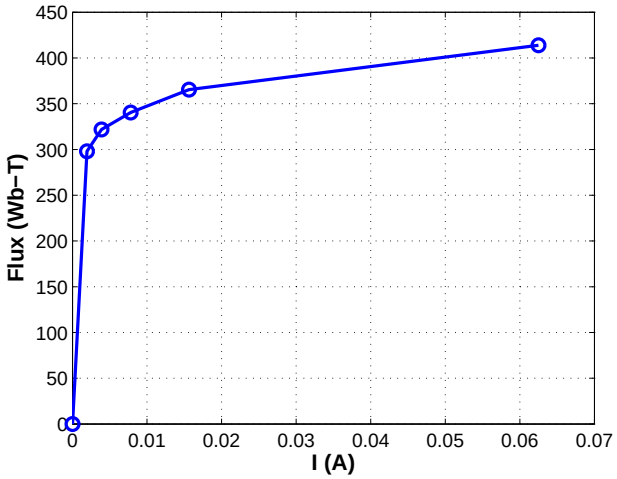


Figure 4.16: Piecewise nonlinear magnetization characteristic of transformer.

4.4 Summary

In this chapter a fully iterative nonlinear electromagnetic transient solver is described for implementation on the FPGA. The nonlinear solver utilizes the compensation method with the Newton-Raphson algorithm, dedicated floating-point arithmetics, sparsity techniques, and parallel Gauss-Jordan elimination as its main modules among others. The entire design is deeply pipelined and parallelled to achieve the highest throughput and lowest latency. The two case studies illustrate the use of both the continuous Newton-Raphson and piecewise Newton-Raphson for the nonlinear solver. Together with other modules developed in Chapter 3, the FPGA-based real-time EMT simulator now includes transmission lines, linear lumped RLCG elements, supply sources, circuit breakers, and nonlinear elements.

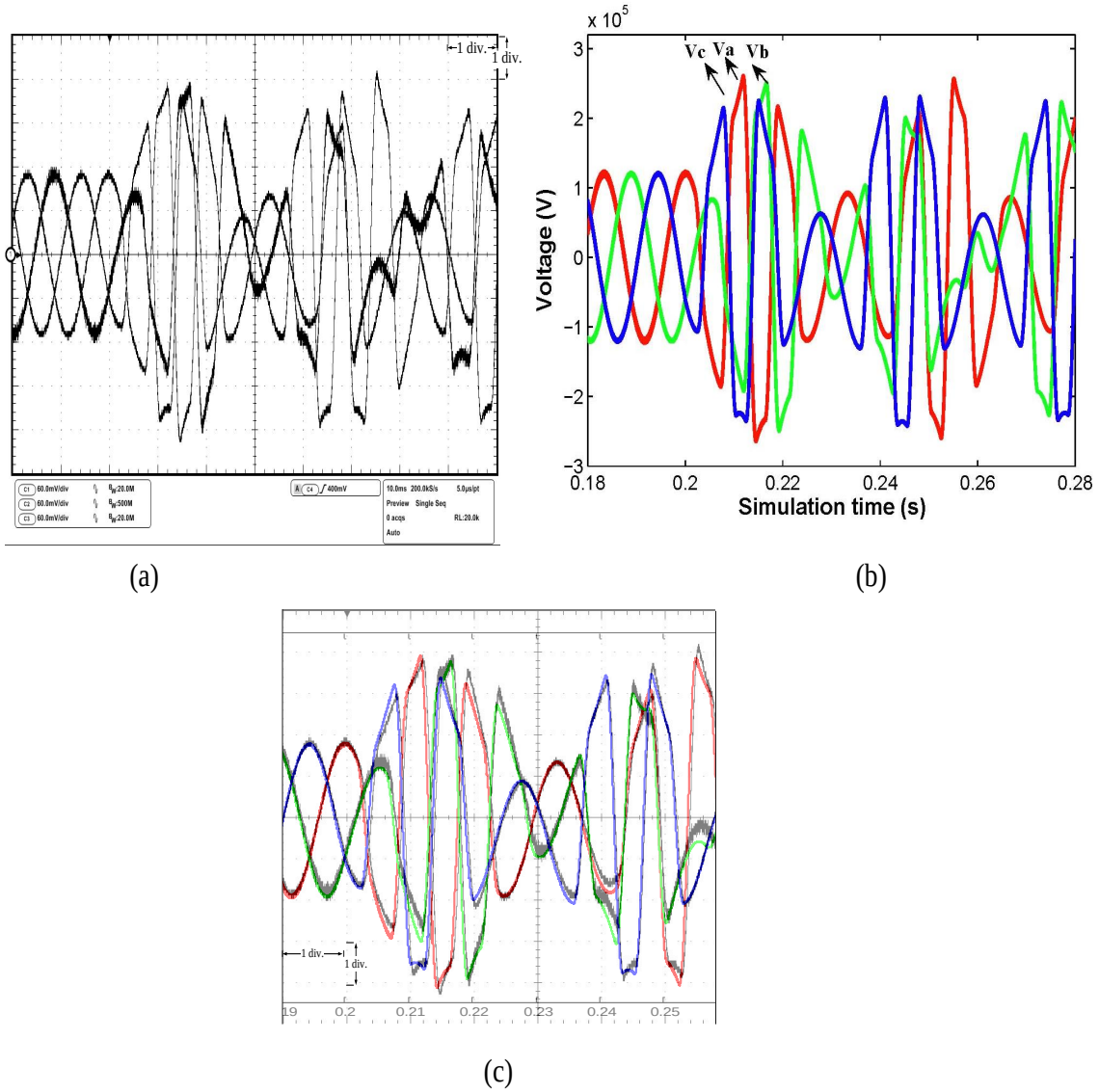
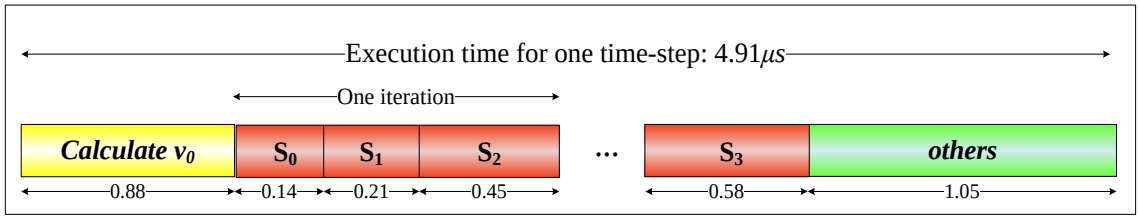
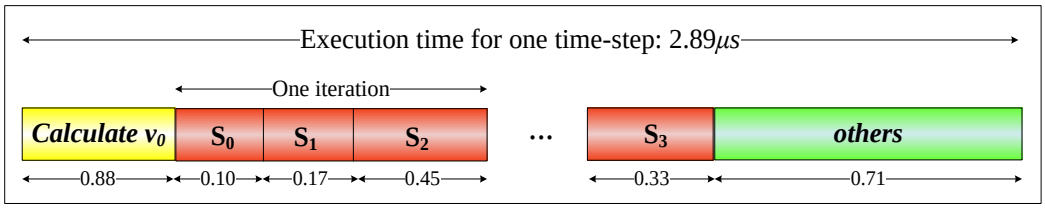


Figure 4.17: Real-time oscilloscope traces (a), off-line simulation results from ATP (b), and zoomed and superimposed view (c) of the three-phase voltages at the transformer terminals during a three-phase-to-ground fault. Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 68kV.



(a) Case Study I



(b) Case Study II

Figure 4.18: Execution time for the case studies in μs. S_i (i=0,..3) are the states of the finite state machine of the nonlinear solver (Fig. 4.5).

5

Digital Hardware Emulation of Universal Machine and Universal Line Models

In this chapter[1], rotating electric machines are emulated in the FPGA using the universal machine model (UM). Moreover, a comprehensive transmission line model called universal line model (ULM) is also implemented in the FPGA. Compared to the frequency-dependent line model (FDLM) implemented in Section 3.1.2, ULM offers more accurate and general simulation of transmission lines and cables to include asymmetrical and unbalance. First an introduction to UM and ULM models is presented. Then the UM and ULM formulation are described, and their detailed hardware implementation details are discussed. A real-time transient simulation case study is used. The captured real-time simulation results have been validated using off-line simulation results from the EMTP-RV software.

5.1 Introduction

Rotating electric machinery are widely used in power systems either as generators or industrial loads, and their accurate modeling is paramount for electromagnetic transient simulation. The machine itself can be of many different types such as induction machine, synchronous machine, and DC machine. Each type of machine may have different electrical representation and custom interfaces between the machine and the network, and between the machine and associated mechanical system. It would be extremely cumbersome and timing-consuming to code each of them individually into the Electromagnetic

[1]Material from this chapter has been published: Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation", *IEEE Trans. on Industrial Electronics*, vol 59, no. 2, pp. 1300-1309, February 2012.

Transient Program (EMTP) [4]. For this reason, the UM model was proposed [64, 65]. It is a unified, generalized model which can be used to represent up to 12 types of rotating machines. The UM model is established on the concept that all types of machines can be represented by some coupled electrical coils, and that the associated mechanical system can be replaced by the equivalent electrical analog.

The propagation of electromagnetic transients on a transmission line is greatly influenced by the frequency dependence of its parameters namely its series impedance and shunt admittance. Traditionally, the most accurate transmission line model is the FDLM built in the modal-domain [56]. The transformation matrix between the modal-domain and the phase-domain is real and constant. This model works well for symmetrical and transposed lines; however, for asymmetrical and untransposed line configurations, the use of a constant modal transformation matrix causes error and numerical instability, especially for cable models. Although this deficiency can be overcome by taking the frequency-dependence of the transformation matrix into account [66], the practical implementation of the model is complicated. Phase-domain modeling avoids the transformation matrix by formulating the transmission line equations directly in the phase-domain [67–69]. Among these phase-domain line models, the ULM [69] is considered numerically efficient and robust for both overhead lines and underground cables.

Accurate modeling of rotating machines and transmission lines are two of the most computationally demanding tasks for a real-time simulator. Moreover, a realistic reproduction of high-frequency transients often requires a very small simulation time-step. Existing real-time simulators largely employ sequential processors such as general purpose CPUs or DSPs as their core computational processors. To meet the stringent real-time step-size constraints, a compromise is usually made between the size of the system simulated and the complexity of the component models. For example, a rotating machine is modeled by its Thévenin equivalent or a low-order lumped machine model, and a transmission line is modeled by a relatively simpler distributed parameter traveling wave model instead of a frequency-dependent model, when large systems need to be simulated in real-time. Of course, sacrificing the model complexity degrades the accuracy of the transients the real-time simulator is able to reproduce. Owing to the fast developments in capacity and speed of high-end FPGA real-time digital hardware emulation of UM and ULM model is possible.

5.2 Universal Machine Model

5.2.1 UM Model Formulation

The UM model is established on the synchronously rotating *dq0* reference frame. It can have at most three stator windings, any number of windings $D_1, D_2, ..., D_m$ on the rotor direct axis (d-axis), and any number of windings $Q_1, Q_2, ..., Q_n$ on the rotor quadrature axis

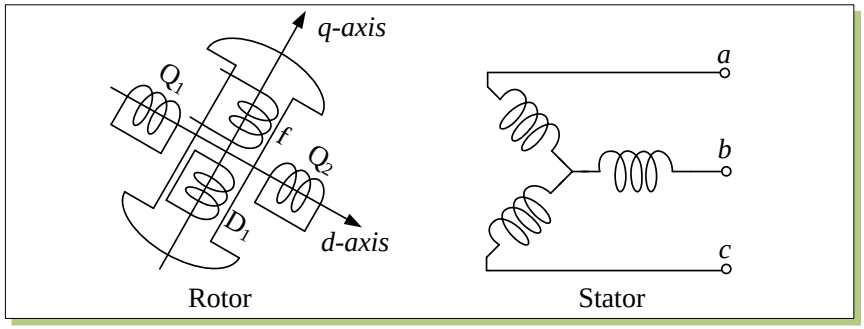


Figure 5.1: Winding representations in the UM model.

(q-axis). Without loss of generality, as shown in Fig. 5.1, three stator windings $\{a, b, c\}$, one field winding f, one damper winding D_1 on d-axis, and two damper windings $\{Q_1, Q_2\}$ on the q-axis are considered in this design. The voltage-current-flux relationship in these coupled windings can be described by

$$\boldsymbol{v}_{dq0} = -\boldsymbol{R}\boldsymbol{i}_{dq0} - \frac{d\boldsymbol{\lambda}_{dq0}}{dt} + \boldsymbol{u}, \tag{5.1}$$

and

$$\boldsymbol{\lambda}_{dq0} = \boldsymbol{L}\boldsymbol{i}_{dq0}, \tag{5.2}$$

where

$$\boldsymbol{v}_{dq0} = \begin{bmatrix} v_d \\ v_q \\ v_0 \\ v_f \\ 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{i}_{dq0} = \begin{bmatrix} i_d \\ i_q \\ i_0 \\ i_f \\ i_{D1} \\ i_{Q1} \\ i_{Q2} \end{bmatrix}, \boldsymbol{\lambda}_{dq0} = \begin{bmatrix} \lambda_d \\ \lambda_q \\ \lambda_0 \\ \lambda_f \\ \lambda_{D1} \\ \lambda_{Q1} \\ \lambda_{Q2} \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} -\omega\lambda_q \\ \omega\lambda_d \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{5.3}$$

are vectors of voltages, currents, flux linkages, and speed voltages of the windings.

$$\boldsymbol{R} = \begin{bmatrix} R_d & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & R_q & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & R_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & R_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & R_{D1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & R_{Q1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & R_{Q2} \end{bmatrix}, \tag{5.4}$$

is a diagonal matrix of winding resistances, while

Table 5.1: Equivalence between mechanical and electrical quantities for the UM model

Mechanical	Electrical
T_m: load torque [Nm]	I_m: current source [A]
T_e: air gap torque [Nm]	I_e: current [A]
J: inertia [kg-m^2]	C: capacitance [F]
D: damping [Nm-s/rad]	$1/R$: conductance [S]
ω: rotor speed [rad/s]	V: voltage [V]

$$\boldsymbol{L} = \begin{bmatrix} L_d & 0 & 0 & M_{df} & M_{dD1} & 0 & 0 \\ 0 & L_q & 0 & 0 & 0 & M_{qQ1} & M_{qQ2} \\ 0 & 0 & L_0 & 0 & 0 & 0 & 0 \\ M_{df} & 0 & 0 & L_f & M_{fD1} & 0 & 0 \\ M_{dD1} & 0 & 0 & M_{fD1} & L_{D1} & 0 & 0 \\ 0 & M_{qQ1} & 0 & 0 & 0 & L_{Q1} & M_{Q1Q2} \\ 0 & M_{qQ2} & 0 & 0 & 0 & M_{Q1Q2} & L_{Q2} \end{bmatrix},$$

is a symmetrical matrix of the winding leakage inductances with L and M denoting the self and mutual inductances, respectively.

The electromagnetic torque is calculated using

$$T_e = \lambda_d i_q - \lambda_q i_d. \tag{5.5}$$

Instead of a mechanical model of the mass-shaft system in most machine models, the UM uses an equivalent electric network with lumped R, L, C elements to represent the mechanical part of machine [64]. This gives the greatest flexibility to model different mechanical systems using the existing electrical models in the EMTP framework. By replacing the mechanical parameters with their analog electrical quantities as shown in Table 5.1, the differential equation (5.6) describing the dynamics of the rotor can be modeled by an electrical circuit. Fig. 5.2 shows an example of a single-mass mechanical system and its electrical analog.

$$T_m = J\frac{d\omega}{dt} + D\omega + T_e. \tag{5.6}$$

5.2.2 Interfacing UM Model with EMTP

To interface the UM model to the existing EMTP framework, the differential equation (5.1) is discretized using the Trapezoidal rule of integration to yield

$$\boldsymbol{v}_{dq0}(t) = -\boldsymbol{R}\boldsymbol{i}_{dq0}(t) - \frac{2}{\Delta t}\boldsymbol{\lambda}_{dq0}(t) + \boldsymbol{u}(t) + \boldsymbol{v}_{hist}, \tag{5.7}$$

where Δt is the time-step and $\boldsymbol{v}_{hist}$ is the history term expressed as

$$\begin{aligned} \boldsymbol{v}_{hist} &= -\boldsymbol{v}_{dq0}(t-\Delta t) - \boldsymbol{R}\boldsymbol{i}_{dq0}(t-\Delta t) \\ &\quad + \frac{2}{\Delta t}\boldsymbol{\lambda}_{dq0}(t-\Delta t) + \boldsymbol{u}(t-\Delta t). \end{aligned} \tag{5.8}$$

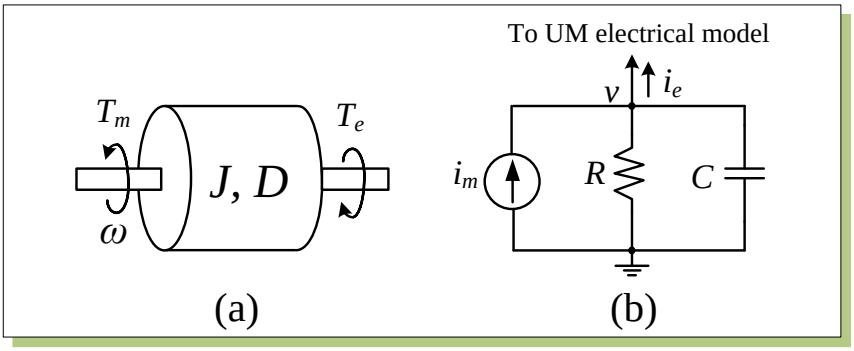


Figure 5.2: (a) Mechanical system of rotor, and (b) its electrical analog.

To incorporate the UM model into the EMTP solution, the compensation method is employed. The network is first solved without the machine and represented by a Thévenin impedance $\boldsymbol{R}_{eq}$ and a voltage source $\boldsymbol{v}_{abc\_0}$ expressed as

$$\boldsymbol{v}_{abc} = \boldsymbol{v}_{abc\_0} + \boldsymbol{R}_{eq}\boldsymbol{i}_{abc}. \tag{5.9}$$

To link the *dq0* quantities of the machine with the *abc* phase quantities of the rest of the network, the Park's transformation matrix $\boldsymbol{P}$ is used. $\boldsymbol{P}$ is an orthogonal matrix defined as

$$\boldsymbol{P} = \sqrt{\frac{2}{3}} \begin{bmatrix} cos(\beta) & cos(\beta - 120^0) & cos(\beta + 120^0) \\ sin(\beta) & sin(\beta - 120^0) & sin(\beta + 120^0) \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}, \tag{5.10}$$

where β is the rotor angle.

Transforming (5.9) into *dq0* frame, we obtain

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \end{bmatrix} = \begin{bmatrix} v_{d\_0} \\ v_{q\_0} \\ v_{0\_0} \end{bmatrix} + \boldsymbol{R}_{eq\_dq} \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix}, \tag{5.11}$$

where

$$\begin{bmatrix} v_{d\_0} \\ v_{q\_0} \\ v_{0\_0} \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} v_{a\_0} \\ v_{b\_0} \\ v_{c\_0} \end{bmatrix}, \tag{5.12}$$

and

$$\boldsymbol{R}_{eq\_dq} = \boldsymbol{P}\boldsymbol{R}_{eq}\boldsymbol{P}^{-1}. \tag{5.13}$$

Substituting (5.11) into (5.7), the voltages are eliminated as

$$\boldsymbol{A}\boldsymbol{i}_{dq0} = \boldsymbol{b}, \tag{5.14}$$

where

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{R}_c + \boldsymbol{R}_{eq\_dq1} + \omega\boldsymbol{L}_2, \\ \boldsymbol{b} &= \boldsymbol{v}_{dq0\_0} + \boldsymbol{v}_{hist}, \end{aligned} \tag{5.15}$$

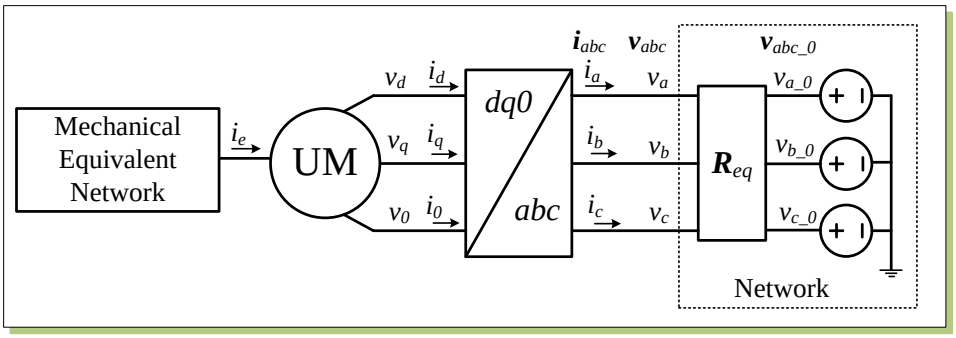


Figure 5.3: Interfacing of the UM model to the network using the compensation method.

with

$$\boldsymbol{R}_c = \boldsymbol{R} + \frac{2}{\Delta t}\boldsymbol{L}, \tag{5.16}$$

$$\boldsymbol{R}_{eq\_dq1} = \begin{bmatrix} [\boldsymbol{R}_{eq\_dq}]_{3\times3} & \boldsymbol{0}_{3\times4} \\ \boldsymbol{0}_{4\times3} & \boldsymbol{0}_{4\times4} \end{bmatrix}, \tag{5.17}$$

$$\boldsymbol{L}_2 = \begin{bmatrix} 0 & L_q & 0 & 0 & 0 & M_{qQ1} & M_{qQ2} \\ -L_d & 0 & 0 & -M_{df} & -M_{dD1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.18}$$

After the machine terminal currents i_d, i_q, i_0 are solved in (5.14), they are transformed back to i_a, i_b, i_c in the *abc* frame and superimposed into the rest of the network to calculate the compensated voltages $\boldsymbol{v}_c$. Fig. 5.3 illustrates this compensation method.

5.2.3 Real-Time Hardware Emulation of UM Model

Hardware Architecture and Parallelism

A hardware module `UM` is designed to emulate the UM model in the FPGA. Fig. 5.4 shows the symbol of this module and its input/output signals. As shown in Fig. 5.5, `UM` module mainly consists of six functional units. The `Speed & Angle` unit is responsible for predicting and calculating rotor speed ω and rotor angle β. The `FrmTran` unit transforms the UM quantities between the *abc* and *dq0* frames. The `Comp`$\boldsymbol{i}_{dq0}$ unit is used to calculate the machine current $\boldsymbol{i}_{dq0}$. The flux linkages $\boldsymbol{\lambda}_{dq0}$ and torque T_e are solved in the `Flux & Torque` unit. The `Update` unit updates machine history terms $\boldsymbol{v}_{hist}$ and the history terms of the equivalent mechanical network. Finally, the `CompVc` unit calculates the complete voltages of the network $\boldsymbol{v}_c$.

To solve the nonlinear equations of the machine, iterations are required. The number of iterations is generally small (1 to 3) due to the relatively large inertia of machine. Fig. 5.6

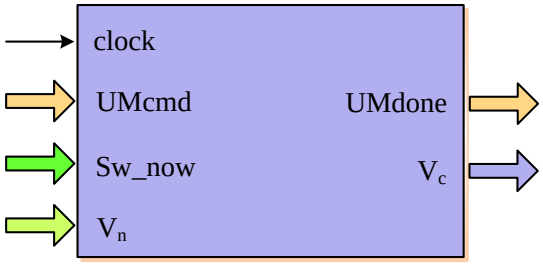


Figure 5.4: `UM` module and its input/output signals.

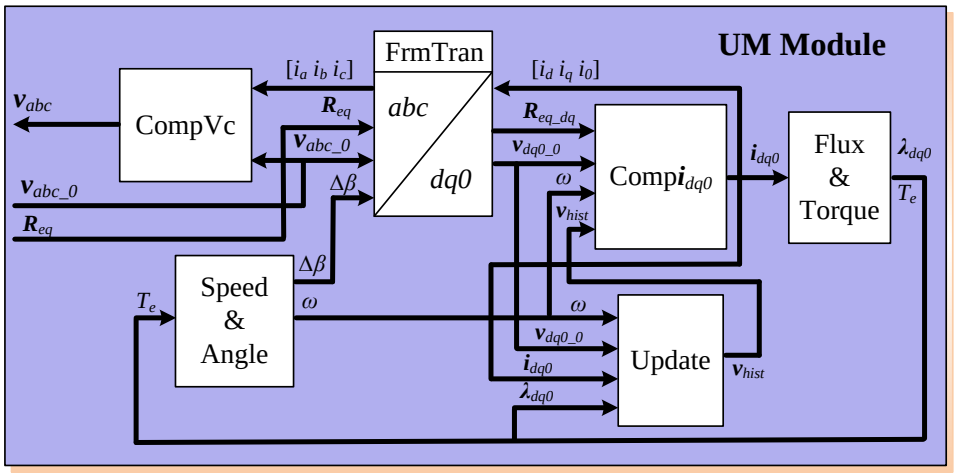


Figure 5.5: Main functional units in the `UM` module.

shows the finite state machine (FSM) diagram of the iteration process. When the network is solved without machines for applying the compensation method [58], the UM procedure starts with the prediction of rotor speed ω using linear extrapolation in state S_1. Then in state S_2 the rotor angle β is calculated, followed by Park's transformation matrices $\boldsymbol{P}$ and $\boldsymbol{P}^{-1}$ (5.10). The $\boldsymbol{P}$ and $\boldsymbol{P}^{-1}$ matrices are calculated using a sinusoidal function look-up table (LUT). The Thévenin equivalent for the rest of network is then transformed into the *abc* frame in state S_3. Then in state S_4 the current $\boldsymbol{i}_{dq0}$ is calculated using a parallel Gauss-Jordan elimination method. Once the $\boldsymbol{i}_{dq0}$ is available, the flux linkages $\boldsymbol{\lambda}_{dq0}$, electromagnetic torque T_e, and rotor speed ω are computed sequentially in *dq0* frame in state S_5. Meanwhile, the $\boldsymbol{i}_{dq0}$ is transformed back to *abc* frame to superimpose into the rest of network to calculate the complete node voltages of the network $\boldsymbol{v}_c$ in state S_5. The calculated ω is compared with the predicted ω in state S_6. If the difference is within the given tolerance, the iteration process is terminated; otherwise the next iteration is started.

Obviously, solving the UM model is very time-consuming since it needs iterations and each iteration involves 6 sequential steps. To improve the hardware computational efficiency of the UM model, parallelism has to be employed. Although the overall procedure is sequential, there exist possible parallel processing paths. For instance, in state S_3 of Fig.

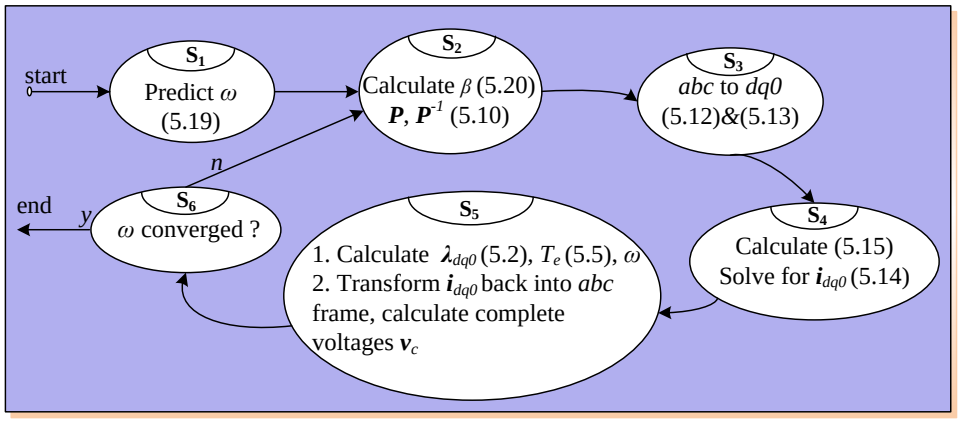


Figure 5.6: Finite state machine diagram of the iteration process of the `UM` module.

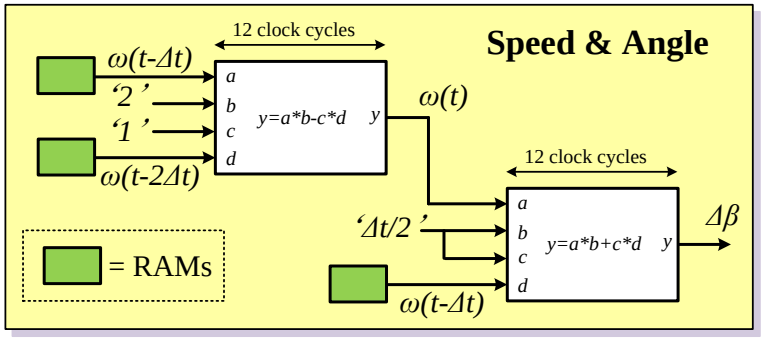


Figure 5.7: Pipelined computation scheme in the `Speed & Angle` unit.

5.6, (5.12) and (5.13) are calculated simultaneously; in state S_4, calculating $\boldsymbol{A}$ and $\boldsymbol{b}$ (5.15) are carried out concurrently; and in state S_5, after $\boldsymbol{i}_{dq0}$ is calculated, the machine variables $\boldsymbol{\lambda}_{dq0}$, T_e, and ω are calculated in *dq0* frame, while $\boldsymbol{i}_{dq0}$ is transformed back to *abc* frame thus the complete voltages $\boldsymbol{v}_c$ are calculated simultaneously. Furthermore, parallel processing can also be applied into each individual calculation as discussed in details as follows.

`Speed & Angle` **Unit**

In the `Speed & Angle` unit, the rotor speed ω is predicted with linear extrapolation as

$$\omega(t) = 2\omega(t - \Delta t) - \omega(t - 2\Delta t). \tag{5.19}$$

The rotor position (incremental) $\Delta\beta$ is then calculated using the predicted speed with trapezoidal rule of integration as

$$\Delta\beta(t) = \beta(t) - \beta(t - \Delta t) = \frac{\Delta t}{2}[\omega(t - \Delta t) + \omega(t)]. \tag{5.20}$$

ω and $\Delta\beta$ are calculated in the `Speed & Angle` unit shown in Fig. 5.7.

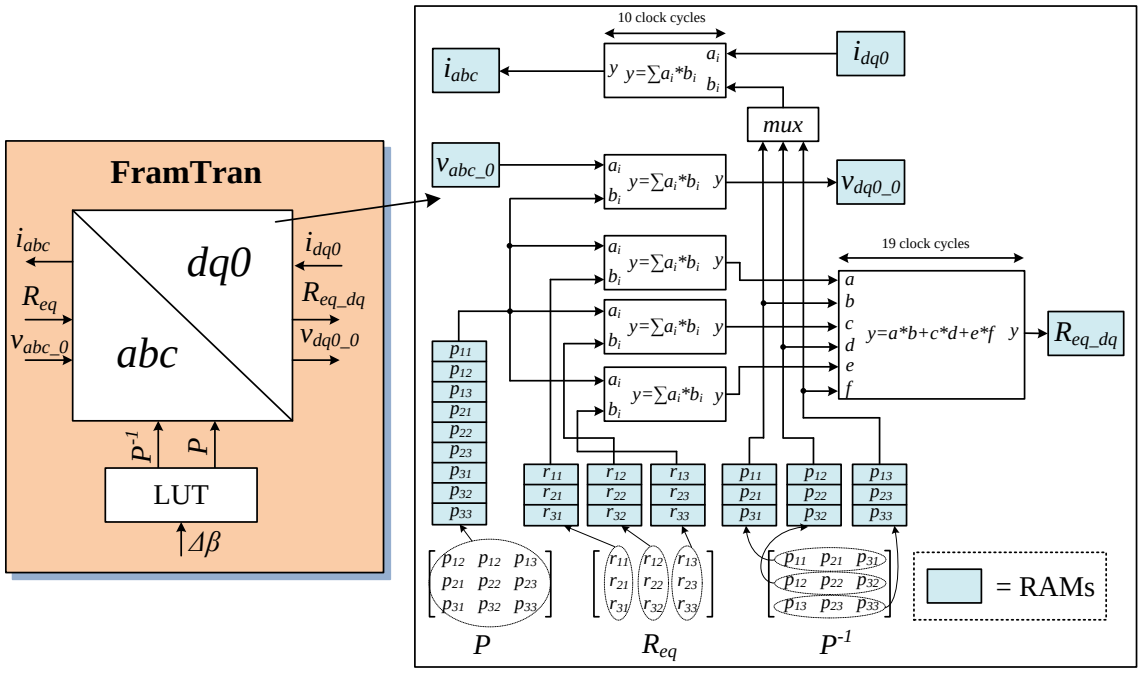


Figure 5.8: Parallel computation scheme in the `FrmTran` unit.

`FramTran` Unit

In the `FrmTran` unit, the quantities in *dq0* and *abc* frame are transformed. This includes transforming $\boldsymbol{v}_{abc\_0}$ into $\boldsymbol{v}_{dq0\_0}$, $\boldsymbol{R}_{eq}$ into $\boldsymbol{R}_{eq\_dq}$ in the *dq0* frame, and $\boldsymbol{i}_{dq0}$ back to $\boldsymbol{i}_{abc}$ in the *abc* frame. Fig. 5.8 shows its parallel computation scheme. First the Park matrices $\boldsymbol{P}^{-1}$ and $\boldsymbol{P}$ are calculated using a LUT based on the calculated $\Delta\beta$ from the `Speed & Angle` unit. The process of the LUT has been discussed in Section 3.3.2. To perform the frame transformation in parallel $\boldsymbol{P}$ and $\boldsymbol{P}^{-1}$ are saved in various RAMs. The matrix $\boldsymbol{P}$ is stored in a single RAM, and $\boldsymbol{P}^{-1}$ is stored in 3 individual RAMs but in a row-wise format, while the matrix $\boldsymbol{R}_{eq}$ is also stored in 3 individual RAMs but in column-wise format. With this configuration the matrix multiplication of $\boldsymbol{P}\boldsymbol{R}_{eq}$ is performed by 3 fast floating-point multiply-accumulate (`FFPMAC`, $y = \sum a_i b_i$) units in parallel. Thus one row of $\boldsymbol{P}$ can multiply three columns of $\boldsymbol{R}_{eq}$ resulting in three elements in one row of the product matrix simultaneously. Then these intermediate results along with three elements in one column of matrix $\boldsymbol{P}^{-1}$ are multiplied using a floating-point multiply-add ($y = a \times d + c \times d + e \times f$) unit resulting in one element of the final product matrix. Meanwhile another `FFPMAC` unit is used to calculate $\boldsymbol{v}_{dq0\_0}$ at the same time. It is worthy to note that this parallel configuration is important to reduce the execution time. For example, performing $\boldsymbol{P}\boldsymbol{R}_{eq}\boldsymbol{P}^{-1}$ sequentially needs 54 clock cycles, while only 9 clock cycles are required in the proposed scheme. The same configuration is used for calculating $\boldsymbol{i}_{abc}$, but a multiplexer is used to select the elements of $\boldsymbol{P}^{-1}$.

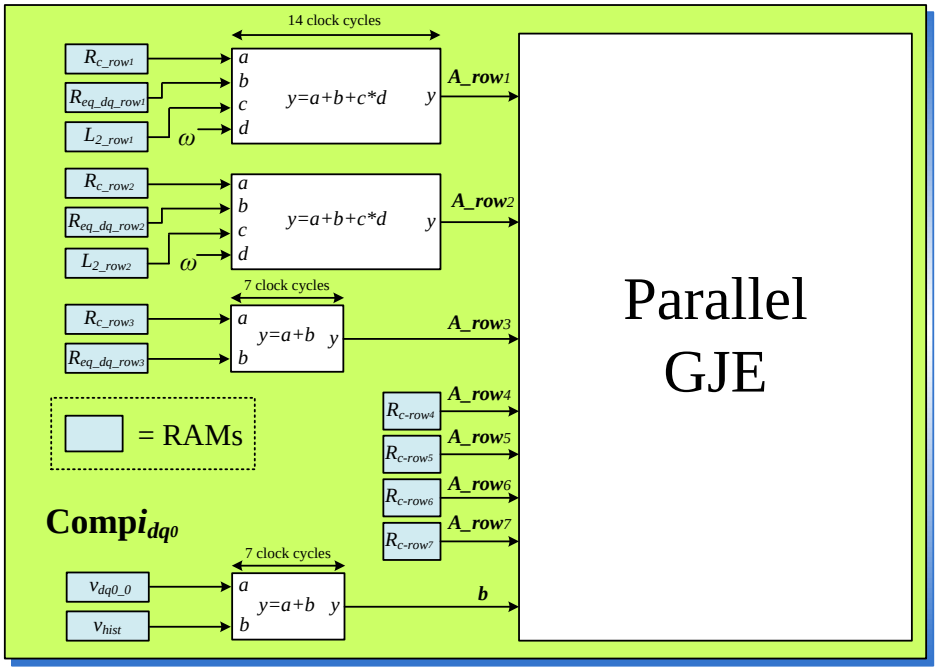


Figure 5.9: Parallel computation scheme in the $\texttt{Comp}\boldsymbol{i}_{dq0}$ unit.

$\texttt{Comp}\boldsymbol{i}_{dq0}$ Unit

In the $\texttt{Comp}\boldsymbol{i}_{dq0}$ unit, a set of algebraic equations (5.14) are solved for $\boldsymbol{i}_{dq0}$. First $\boldsymbol{A}$ and $\boldsymbol{b}$ are calculated (5.15). Since $\boldsymbol{A}$ is a 7×7 matrix for 7 windings representation, 49 clock cycles (not including hardware latency) are required to calculate $\boldsymbol{A}$. This is very time-consuming especially when the number of machines is large. A parallel scheme shown in Fig. 5.9 is designed to improve the computational efficiency. Each row of matrix $\boldsymbol{A}$ is calculated independently. Since only first 2 rows of the $\boldsymbol{L}_2$ (5.18) and first 3 rows of the $\boldsymbol{R}_{eq\_dq1}$ (5.17) have non-zero elements, the first 2 rows of $\boldsymbol{A}$ are calculated by 2 arithmetic ($y = a + b + c \times d$) units, while the third row of $\boldsymbol{A}$ is carried out by an adder. The rest rows of $\boldsymbol{A}$ are from matrix $\boldsymbol{R}_c$ (5.16), so they are retrieved directly from corresponding RAMs. The calculation of $\boldsymbol{b}$ is performed concurrently by an adder. After $\boldsymbol{A}$ and $\boldsymbol{b}$ are available, $\boldsymbol{i}_{dq0}$ are solved in a parallel `GJE` unit which has been discussed in Section 4.2.4. Here 7 independent elimination units are utilized in the parallel `GJE` unit for solving a set of 7 algebraic equations in parallel.

`Flux & Torque` Unit

In the `Flux & Torque` unit, the flux $\boldsymbol{\lambda}_{dq0}$ (5.2) and torque T_e (5.5) are calculated. As can be seen in (5.2), the matrix $\boldsymbol{L}$ is very sparse, so the sparse matrix technique is employed, as shown in Fig. 5.10. After $\boldsymbol{\lambda}_{dq0}$ is available, T_e is calculated.

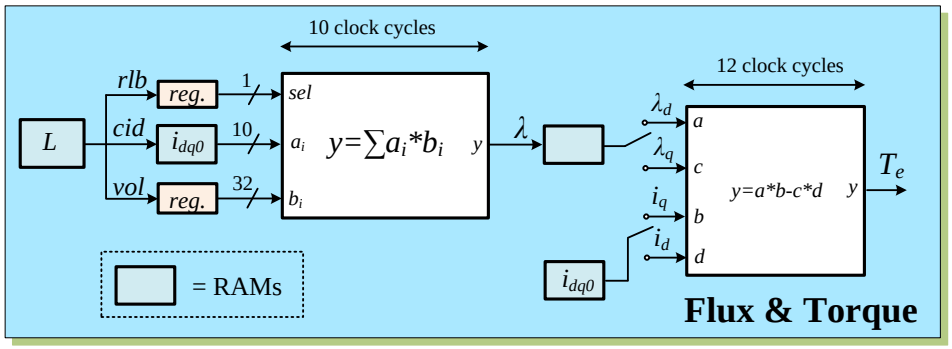


Figure 5.10: Pipelined computation scheme in the `Flux & Torque` unit.

`Update` **Unit**

In the `Update` unit, the history term for machine $\boldsymbol{v}_{hist}$ and for the RLC elements used to represent the equivalent mechanical part of the machine are updated. $\boldsymbol{V}_{hist}$ is updated in the `Update` unit, while the update of RLC elements history has been discussed in Section 3.2.2.

`CompVc` **Unit**

Finally, the compensated node voltages $\boldsymbol{v}_c$ are calculated by superimposing $\boldsymbol{i}_{abc}$ into the linear network. The hardware unit to do it is the same with the `CompVc` module in Section 4.2.5.

5.3 Universal Line Model

5.3.1 ULM Model Formulation in Frequency-Domain

For a n-phase transmission line as shown in Fig. 5.11 (a), the solution of the traveling wave equations can be expressed in frequency-domain at the sending-end ('k') and the receiving-end ('m') as

$$\begin{aligned} \boldsymbol{I}_k &= \boldsymbol{Y}_c\boldsymbol{V}_k - 2\boldsymbol{I}_{ki} = \boldsymbol{Y}_c\boldsymbol{V}_k - 2\boldsymbol{H}\boldsymbol{I}_{mr}, \\ \boldsymbol{I}_m &= \boldsymbol{Y}_c\boldsymbol{V}_m - 2\boldsymbol{I}_{mi} = \boldsymbol{Y}_c\boldsymbol{V}_m - 2\boldsymbol{H}\boldsymbol{I}_{kr}. \end{aligned} \tag{5.21}$$

In the above equations $\boldsymbol{I}_k$, $\boldsymbol{V}_k$ and $\boldsymbol{I}_m$, $\boldsymbol{V}_m$ are n dimensional current and voltage vectors at both line ends, respectively. $\boldsymbol{I}_{ki}$ and $\boldsymbol{I}_{mi}$ are the incident currents, whereas $\boldsymbol{I}_{kr}$ and $\boldsymbol{I}_{mr}$ are the reflected currents. The two $n\times n$ matrix transfer functions which characterize a transmission line are the characteristic admittance matrix $\boldsymbol{Y}_c$ and the propagation matrix $\boldsymbol{H}$ expressed as

$$\boldsymbol{Y}_c = \sqrt{\boldsymbol{Y}/\boldsymbol{Z}}, \tag{5.22}$$

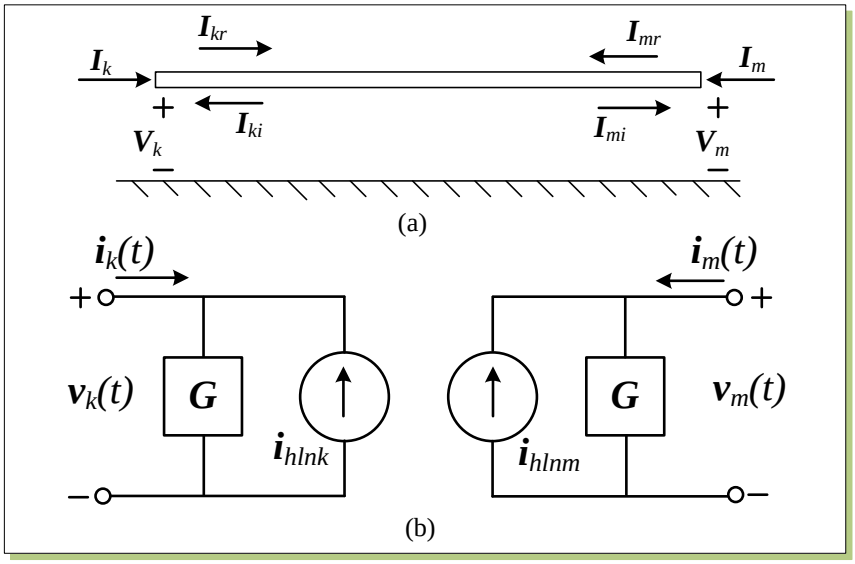


Figure 5.11: (a) An n-phase transmission line, and (b) its time-domain representation.

and

$$\boldsymbol{H} = e^{-\sqrt{(\boldsymbol{YZ})}l}, \tag{5.23}$$

where the $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are $n \times n$ shunt admittance and series impedance matrices per unit length, respectively; l is the line length.

In order to implement the model in time-domain, the elements of $\boldsymbol{Y}_c$ and $\boldsymbol{H}$ are approximated with rational functions. The fitting method used here is *Vector Fitting* (VF) [70]. The elements of $\boldsymbol{Y}_c$ are in general very smooth and can be easily fitted in the phase-domain. The (i, j) element of $\boldsymbol{Y}_c$ is expressed as

$$\boldsymbol{Y}_{c,(i,j)}(s) = \sum_{m=1}^{N_p} \frac{\boldsymbol{r}_{Y_c,(i,j)}(m)}{s - \boldsymbol{p}_{Y_c}(m)} + \boldsymbol{d}_{(i,j)}, \tag{5.24}$$

where N_p is the number of poles; $\boldsymbol{r}_{Y_c}$, $\boldsymbol{p}_{Y_c}$, and $\boldsymbol{d}$ are residues, poles, and proportional terms, respectively. Note that all elements of $\boldsymbol{Y}_c$ have identical poles $\boldsymbol{p}_{Y_c}$.

The fitting of the propagation matrix $\boldsymbol{H}$ is somewhat different. It is first fitted in the modal-domain with poles and time delays in each mode, followed by final fitting in the phase-domain. The (i, j) element of $\boldsymbol{H}$ is expressed as

$$\boldsymbol{H}_{(i,j)}(s) = \sum_{k=1}^{N_g} \left(\sum_{n=1}^{N_{p,k}} \frac{\boldsymbol{r}_{H,(i,j),k}(n)}{s - \boldsymbol{p}_{H,k}(n)} \right) e^{-s\tau_k}, \tag{5.25}$$

where N_g denotes the number of modes; $N_{p,k}$ and τ_k are the number of poles and time delay used for fitting the k^{th} mode. $\boldsymbol{r}_{H,k}$ and $\boldsymbol{p}_{H,k}$ are residues and poles for k^{th} mode. Again the poles are identical for all elements in each mode.

5.3.2 Time-Domain Representation

By transforming (5.21) into the time-domain using inverse Fourier Transformation, the Norton equivalent circuit for the ULM model is obtained, as shown in Fig. 5.11 (b). The history current $\boldsymbol{i}_{hln_k}, \boldsymbol{i}_{hln_m}$ and equivalent impedance matrix $\boldsymbol{G}$ are expressed as

$$\begin{aligned} \boldsymbol{i}_{hln_k} &= \boldsymbol{Y}_c * \boldsymbol{v}_k(t) - 2\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau), \\ \boldsymbol{i}_{hln_m} &= \boldsymbol{Y}_c * \boldsymbol{v}_m(t) - 2\boldsymbol{H} * \boldsymbol{i}_{kr}(t-\tau), \end{aligned} \tag{5.26}$$

and

$$\boldsymbol{G} = \boldsymbol{d} + \boldsymbol{r}_{Y_c}\boldsymbol{\lambda}_{Y_c}, \tag{5.27}$$

where the symbol '*' denotes the matrix-vector convolution. The coefficient $\boldsymbol{\lambda}_{Y_c}$ is defined as

$$\boldsymbol{\lambda}_{Y_c} = (\frac{\Delta t}{2})/(1 - \boldsymbol{p}_{Y_c}\frac{\Delta t}{2}). \tag{5.28}$$

To perform the convolution of $\boldsymbol{Y}_c * \boldsymbol{v}_k(t)$, a state variable $\boldsymbol{x}_{Y_c}$ is defined as

$$\boldsymbol{x}_{Y_c}(t) = \boldsymbol{\alpha}_{Y_c}\boldsymbol{x}_{Y_c}(t-\Delta t) + \boldsymbol{v}_k(t-\Delta t). \tag{5.29}$$

The convolution is then computed using

$$\boldsymbol{Y}_c * \boldsymbol{v}_k(t) = \boldsymbol{c}_{Y_c}\boldsymbol{x}_{Y_c}(t), \tag{5.30}$$

where the coefficient $\boldsymbol{\alpha}_{Y_c}$ and $\boldsymbol{c}_{Y_c}$ are expressed as

$$\boldsymbol{\alpha}_{Y_c} = (1 + \boldsymbol{p}_{Y_c}\frac{\Delta t}{2})/(1 - \boldsymbol{p}_{Y_c}\frac{\Delta t}{2}), \tag{5.31}$$

and

$$\boldsymbol{c}_{Y_c} = \boldsymbol{r}_{Y_c}(\boldsymbol{\alpha}_{Y_c} + 1)\boldsymbol{\lambda}_{Y_c}. \tag{5.32}$$

To perform the convolution of $\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau)$, a state variable $\boldsymbol{x}_H$ is defined as

$$\boldsymbol{x}_H(t) = \boldsymbol{\alpha}_H\boldsymbol{x}_H(t-\Delta t) + \boldsymbol{i}_{mr}(t-\tau-\Delta t). \tag{5.33}$$

The convolution is then computed using

$$\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau) = \boldsymbol{c}_H\boldsymbol{x}_H(t) + \boldsymbol{G}_H i_{mr}(t-\tau), \tag{5.34}$$

where the coefficient $\boldsymbol{\alpha}_H$, $\boldsymbol{c}_H$, and $\boldsymbol{G}_H$ are expressed as

$$\boldsymbol{\alpha}_H = (1 + \boldsymbol{p}_H\frac{\Delta t}{2})/(1 - \boldsymbol{p}_H\frac{\Delta t}{2}), \tag{5.35}$$

$$\boldsymbol{c}_H = \boldsymbol{r}_H(\boldsymbol{\alpha}_H + 1)\boldsymbol{\lambda}_H, \tag{5.36}$$

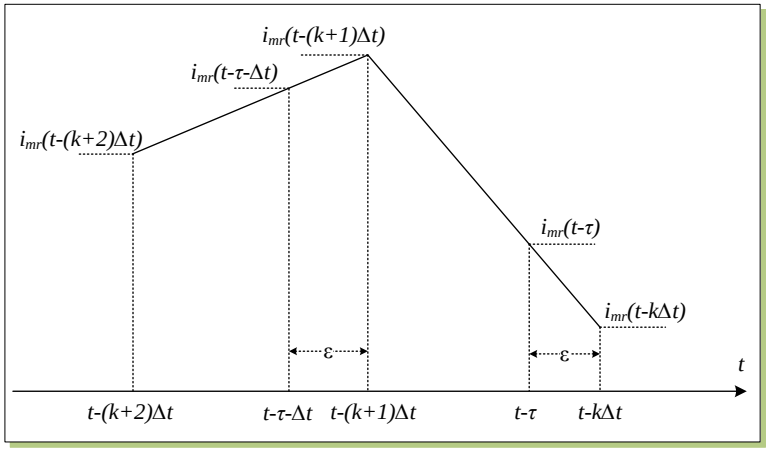


Figure 5.12: Linear interpolation for calculating $\boldsymbol{i}_{mr}(t-\tau)$ and $\boldsymbol{i}_{mr}(t-\tau-\Delta t)$.

and

$$\boldsymbol{G}_H = \boldsymbol{r}_H \boldsymbol{\lambda}_H, \tag{5.37}$$

with

$$\boldsymbol{\lambda}_H = (\frac{\Delta t}{2})/(1 - \boldsymbol{p}_H \frac{\Delta t}{2}). \tag{5.38}$$

As seen from (5.33) and (5.34), $\boldsymbol{i}_{mr}(t-\tau-\Delta t)$ and $\boldsymbol{i}_{mr}(t-\tau)$ are required to perform the convolution of $\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau)$. Linear interpolation is used to accurately calculate $\boldsymbol{i}_{mr}(t-\tau-\Delta t)$ and $\boldsymbol{i}_{mr}(t-\tau)$ since the travel time τ is normally not an integer multiple of the time-step Δt. Assume that τ is expressed as:

$$\tau = (k+\varepsilon)\Delta t, \tag{5.39}$$

where k is an integer and ε is a number between 0 and 1.

As illustrated in Fig. 5.12, $\boldsymbol{i}_{mr}(t-\tau-\Delta t)$ and $\boldsymbol{i}_{mr}(t-\tau)$ can be calculated using linear interpolation as

$$\boldsymbol{i}_{mr}(t-\tau-\Delta t) = \boldsymbol{i}_{mr}(t-(k+2)\Delta t) + (\boldsymbol{i}_{mr}(t-(k+1)\Delta t) - \boldsymbol{i}_{mr}(t-(k+2)\Delta t))(1+\varepsilon), \tag{5.40}$$

and

$$\boldsymbol{i}_{mr}(t-\tau) = \boldsymbol{i}_{mr}(t-(k+1)\Delta t) + (\boldsymbol{i}_{mr}(t-k\Delta t) - \boldsymbol{i}_{mr}(t-(k+1)\Delta t))(1+\varepsilon). \tag{5.41}$$

In order to carry out these two interpolations $(k+2)$ time-steps history values of $\boldsymbol{i}_{mr}(t)$ need to be saved. For example, if $\tau = 9.26\Delta t$, 11 time-steps history values of $\boldsymbol{i}_{mr}(t)$ must be saved.

5.3.3 Real-Time Hardware Emulation of ULM Model

Hardware Architecture and Parallelism

A hardware module `ULM` is designed in the FPGA to emulate the ULM model in real time. Fig. 5.13 shows the symbol of the `ULM` module and its input/output signals. As seen from

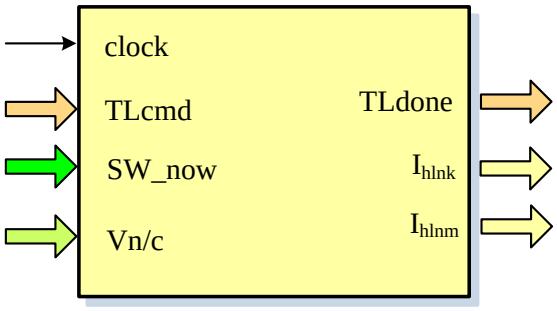


Figure 5.13: `ULM` module and its input/output signals.

the (5.26) the main computations involved in the ULM model are two convolutions at both ends of the transmission line. Depending on the number of phases of the transmission line and the number of poles of fitted rational functions, the convolution is in general quite expensive computationally. Parallel computation is essential to improve the computational efficiency. First, due to the traveling time delay of the ULM model, the sending-end and receiving-end of a transmission line are decoupled. Thus, the calculations at both ends can be conducted fully in parallel. Further more, the two convolutions can be performed at different stage of a simulation time-step. The convolution of $\boldsymbol{H} * \boldsymbol{i}_{mr}(t-\tau)$ is independent of the node voltages, so that it can be carried out before the network is solved; whereas the convolution of $\boldsymbol{Y}_c * \boldsymbol{v}_k(t)$ must be processed after network is solved. In this way the computation time can be reduced significantly. The computational efficiency can be improved even further for multi-phase transmission lines where the calculation in each phase can be executed simultaneously. Based on these three considerations, the detailed hardware computation units of the `ULM` module are designed, as shown in Fig. 5.14. It can first be seen that two identical parts are implemented, for sending-end and receiving-end of transmission line, respectively. To explain these computation units easily, rearrange (5.26), (5.30), and (5.34) to obtain

$$\boldsymbol{i}_{hln_k} = \boldsymbol{i}_{hln_{k1}} - 2(\boldsymbol{i}_{hln_{k2}} + \boldsymbol{i}_{hln_{k3}}), \tag{5.42}$$

where

$$\boldsymbol{i}_{hln_{k1}} = \boldsymbol{c}_{Y_c}\boldsymbol{x}_{Y_c}(t), \tag{5.43}$$

$$\boldsymbol{i}_{hln_{k2}} = \boldsymbol{c}_H\boldsymbol{x}_H(t), \tag{5.44}$$

$$\boldsymbol{i}_{hln_{k3}} = \boldsymbol{G}_H i_{mr}(t-\tau). \tag{5.45}$$

The `Interpolation` unit is responsible for calculating $\boldsymbol{i}_{mr}(t-\tau-\Delta t)$ (5.40) and $\boldsymbol{i}_{mr}(t-\tau)$ (5.41). The `Update` $\boldsymbol{x}$ unit is used to update two state variables $\boldsymbol{x}_{Y_c}(t)$ (5.29) and $\boldsymbol{x}_H(t)$ (5.33). The `Convolution` unit is responsible for calculating $\boldsymbol{i}_{hln_{k1}}$ (5.43) and $\boldsymbol{i}_{hln_{k2}}$ (5.44). A fast floating-point multiply-accumulator (`FFPMAC`, $y = \sum a_i b_i$) unit is used to calculate $\boldsymbol{i}_{hln_{k3}}$ (5.45). Finally the line history term $\boldsymbol{i}_{hln_k}$ is obtained (5.42) in the `Ihlnk` unit.

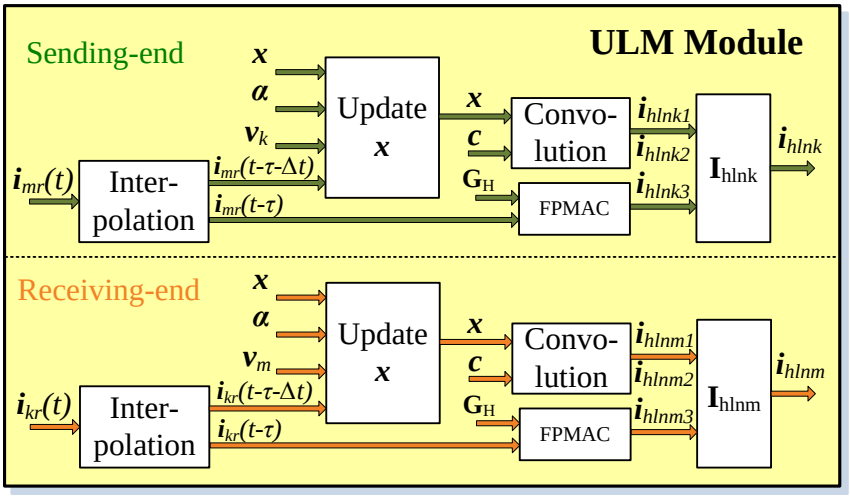


Figure 5.14: Main functional units implemented in the `ULM` module showing parallel computations.

`Update` $\boldsymbol{x}$ Unit

Fig. 5.15 illustrates the parallel computation scheme of (5.29) or (5.33) in the `Update` $\boldsymbol{x}$ unit for a three-phase transmission line. The signals shown in this figure are for (5.29) only. Here the state variable $\boldsymbol{x}_{Y_c}$ and coefficient $\boldsymbol{\alpha}_{Y_c}$ are $N_p \times 3$ matrices, whereas the line voltage $\boldsymbol{v}_k$ is a 1×3 vector. Each column corresponds to one phase denoted by subscript {a,b,c}. All matrices are stored in 3 individual RAMs in column-wise format. The $\boldsymbol{x}_{Y_c}$ is calculated using three floating-point multiply-add ($y = a \times b + c$) units for three phases in parallel. The updated $\boldsymbol{x}_{Y_c}$ is sent back to the RAMs which are dual-port that support the 'read' and 'write' functions simultaneously.

`Convolution` Unit

Once $\boldsymbol{x}_{Y_c}$ or $\boldsymbol{x}_H$ is updated, the convolution of (5.43) or (5.44) can be carried out. Fig. 5.16 illustrates the parallel computation scheme in the `Convolution` unit for a three-phase transmission line. The signals shown in this figure are for (5.43) only. Here the coefficient $\boldsymbol{c}_{Y_c}$ is a $3N_p \times 3$ matrix and stored in three RAMs in column-wise format. Three `FFPMAC` ($y = \sum a_i b_i$) units are used for calculations in the three phases in parallel, when a connected floating-point add ($y = a + b + c$) unit combines them to get final result of convolution. Note that this final result is not available for three phases at the same time. It comes sequentially as phase a, phase b, and phase c. Full parallelism is possible if more arithmetic units are designed subject to FPGA resource utilization.

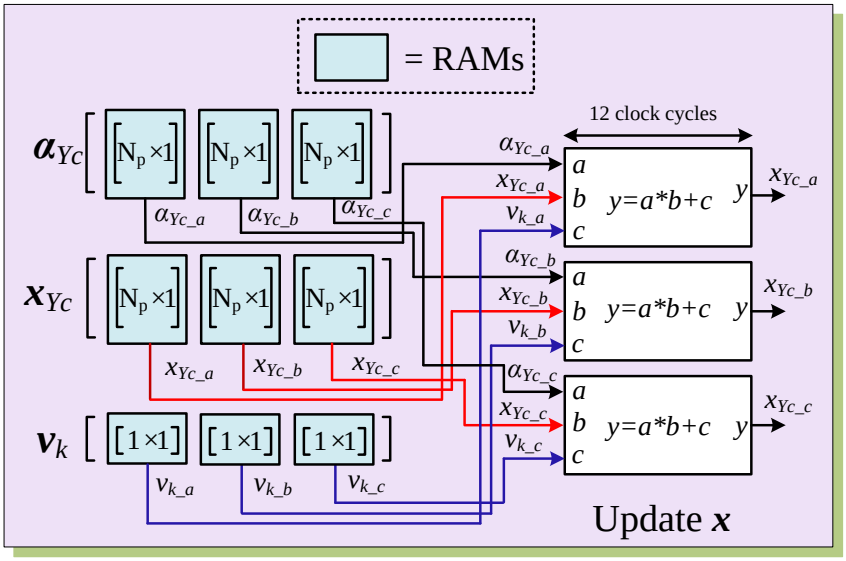


Figure 5.15: Parallel computation scheme in `Update` $\boldsymbol{x}$ unit.

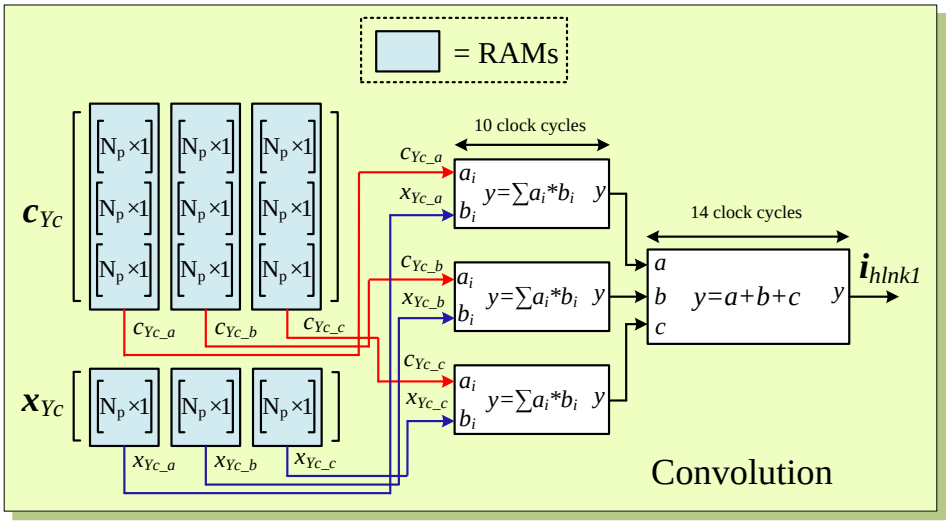


Figure 5.16: Parallel computation scheme in `Convolution` unit.

`Interpolation` Unit

In the `Interpolation` unit, $\boldsymbol{i}_{mr}(t-\tau-\Delta t)$ (5.40) and $\boldsymbol{i}_{mr}(t-\tau)$ (5.41) are calculated. The hardware design of this unit has been shown in Section 4.2.2.

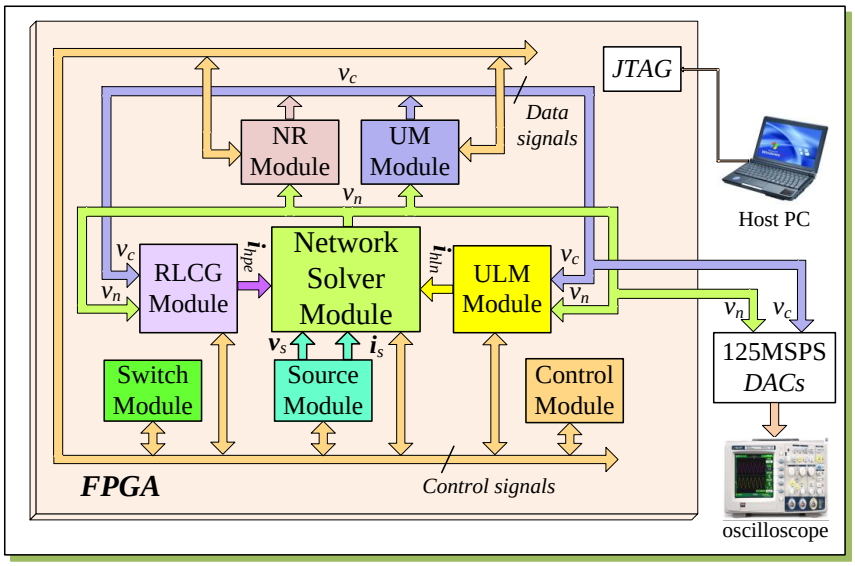


Figure 5.17: Overall hardware architecture of the real-time network emulator.

5.4 Network Hardware Emulation

5.4.1 Hardware Architecture and Parallelism

The hardware emulation of the ULM and UM models along with the network solution is realized on a single FPGA platform. The UM and ULM models are implemented in the `UM` module and `ULM` module respectively as shown in Fig. 5.5 and Fig. 5.14. The remaining modules that assist in the network solution include: (a) `Source` module; (b) `RLCG` module; (c) `NR` module; (d) `Switch` module; (e) `Network Solver` module; and (f) `Control` module. These modules have been implemented in the FPGA as presented in Chapter 3 and 4. The overall hardware architecture is shown in Fig. 5.17. Fig. 5.18 shows more detailed functional units inside each module.

Fig. 5.19 shows the overall procedure in a simulation time-step in the proposed hardware. There are 4 stages in a simulation time-step. In *Stage 1*, the `Source` module calculates the known voltage and current sources for the `Network Solver` module; the `RLCG` module and `ULM` module send out their corresponding history current terms to the `Network Solver` module; whereas the `Switch` module checks the switches state and sends it to the `Network Solver` module. During *Stage 2*, the node voltages are solved without taking into account the electric machines and nonlinear elements. In *Stage 3*, the `UM` module and `NR` module starts to solve machines' equations and nonlinear equations, respectively, and the complete node voltages are calculated. Meanwhile, the `ULM` module computes the convolution for $\boldsymbol{i}_{hlnk_1}$ and $\boldsymbol{i}_{hlnk_3}$ (updating history term, part a). Finally, the upgrading of history terms in the `RLCG` module, `UM` module, `NR` module, and the calculation of $\boldsymbol{i}_{hlnk_2}$ and $\boldsymbol{i}_{hlnk}$ (updating history term, part b) in the `ULM` module are carried out in

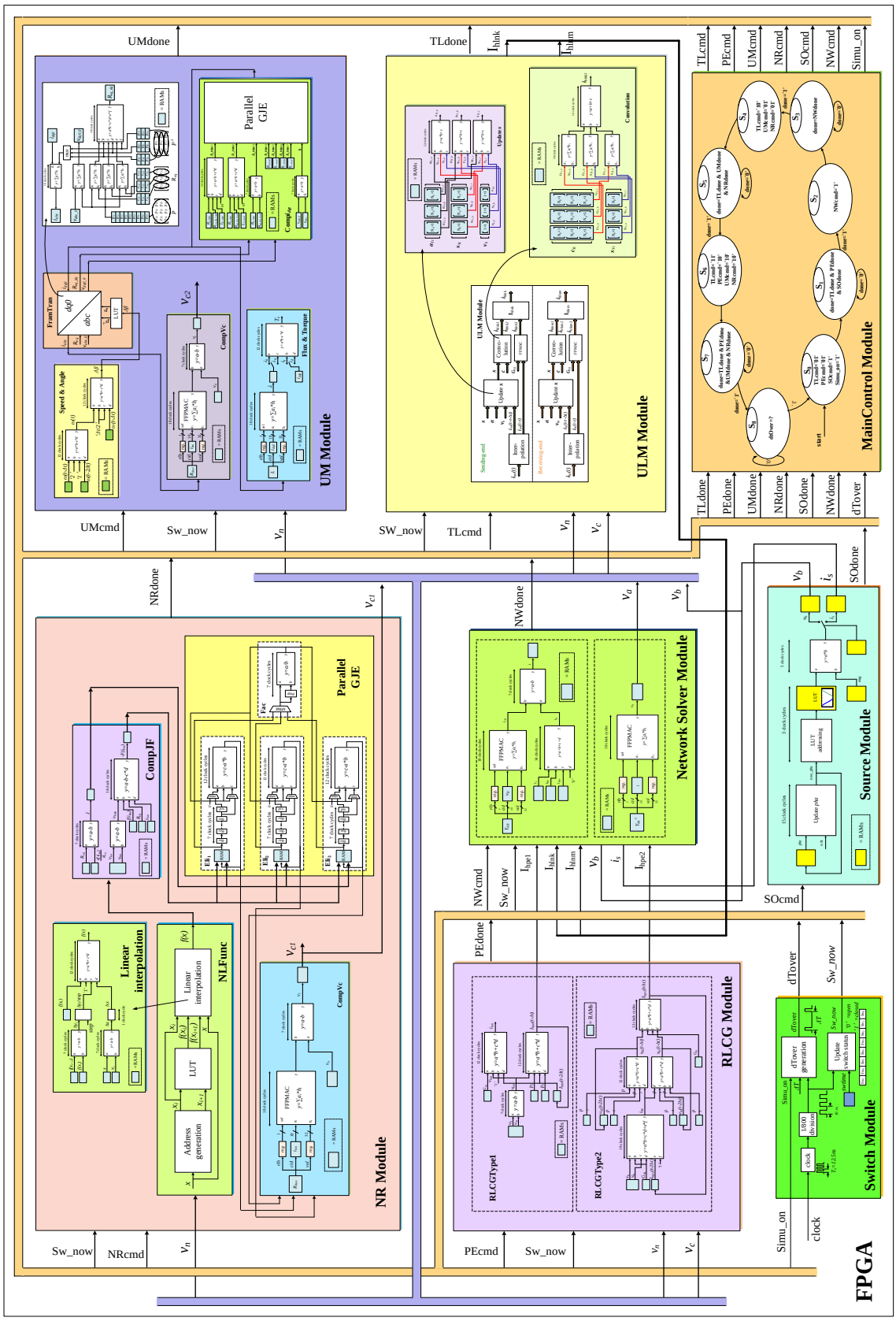


Figure 5.18: Detailed functional units of the real-time network emulator.

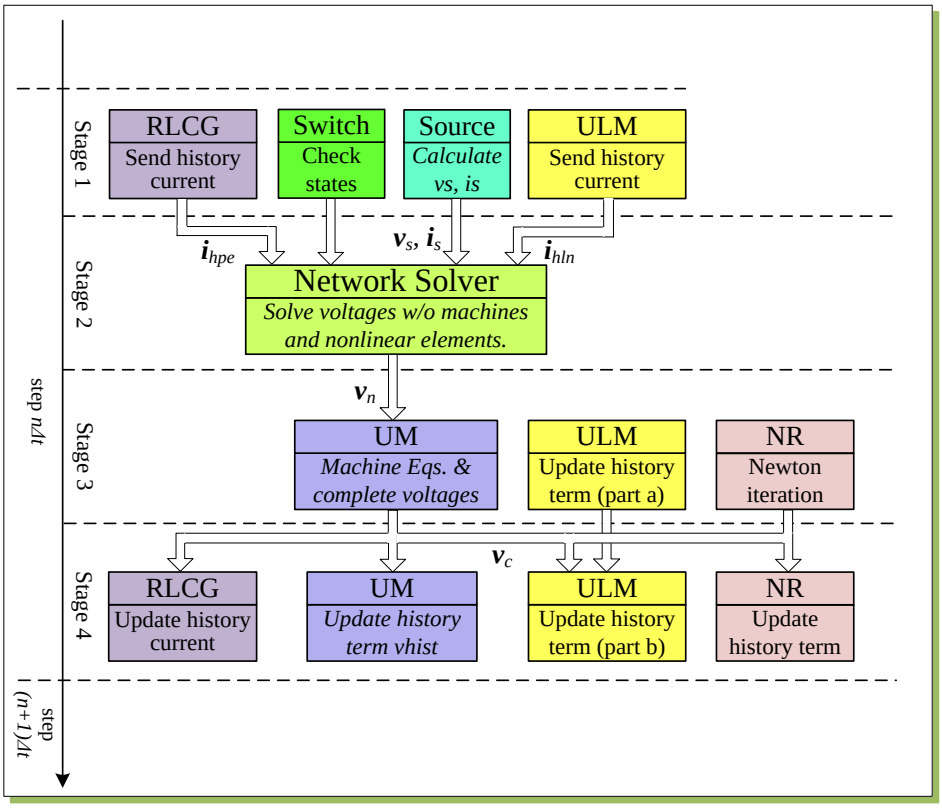


Figure 5.19: Operations within one time-step of the real-time network emulator.

Stage 4. From the above procedure, it is obvious that the parallel processing exists in each stage, while preserving the necessary sequential stages in the overall transient simulation algorithm.

5.4.2 FPGA Resource Utilization

The FPGA platform used to implement the real-time electromagnetic transient network emulator is a Altera Stratix III Development Board DE3, as shown in Fig. 4.11. Fig. 5.20 shows the FPGA hardware resource utilization generated by the Altera Quartus II software in percentage for each module of the real-time network emulator. As can be seen from this figure, the `ULM` module utilizes the most logic resource of the FPGA, and the `UM` module has the second highest resource consumption in the FPGA.

5.5 Real-Time Simulation Case Study

To show the effectiveness of proposed hardware design, an example power system is simulated. The system consists of 3 UM synchronous generators, 2 ULM lines, 3 transformers, and 3 loads as shown in Fig. 5.21. The complete system data is listed in Appendix C. The

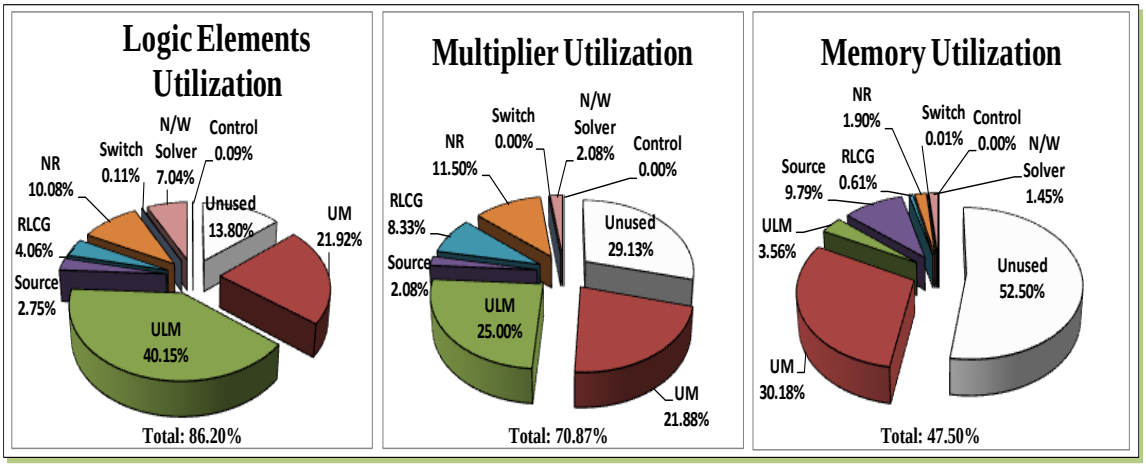


Figure 5.20: FPGA resources utilized by modules of the real-time network emulator.

transformers are modeled as equivalent leakage reactance only. Each ULM utilizes 9th order rational functions to fit its characteristic admittance matrix and propagation matrix.

Two transients events are simulated. The first transient event is a three-phase-to-ground fault at Bus 3 which occurs at $t = 0.1s$. Fig. 5.22 (a) shows the three-phase voltages waveforms at Bus 2 captured from a real-time oscilloscope connected to the DACs. Identical behavior can be observed from Fig. 5.22 (b) which shows the off-line EMTP-RV simulation with a time-step of 8 μs and (c) which shows the zoomed and superimposed view. The second transient event is the capacitor C switched at Bus 3 at time $t = 0.1s$. Fig. 5.23 (a) and Fig. 5.24 (a) show the three-phase voltages waveforms at Bus 3 and electromagnetic torque of UM_2 from the real-time oscilloscope. Fig. 5.23 (b) and Fig. 5.24 (b) show the corresponding transient waveforms obtained from the off-line simulation. Again, detailed agreement between EMTP-RV off-line simulation and the real-time electromagnetic transient network emulator results can be observed. Due to the limited 14-bit resolution of the DACs, small discrepancies exist such as the initial low frequency torque oscillation seen in Fig. 5.24 (b) (off-line simulation) which is truncated from Fig. 5.24 (a) (real-time simulation). Nevertheless, when the transient data in 32-bit floating-point format is stored in the memory and plotted, the low frequency torque oscillation is visible as shown in Fig. 5.24 (c).

The achieved time-step in this implementation is 6.72μs based on a FPGA clock frequency of 130MHz. A detailed break down of this time-step for various stages of computation (Fig. 5.19) and modules is shown in Fig. 5.25. The computation for the UM equations in *Stage 3* is most time-consuming, and the updating of history terms in the `ULM` module has the second highest latency. The detailed execution time of one UM generator and one ULM line in *Stage 3* are also shown in Fig. 5.25. The computation of the 3 UM generators in the system is pipelined through one `UM` module, while the computation of the 2 ULM lines is pipelined through one `ULM` module. As seen from Fig. 5.20, the FPGA

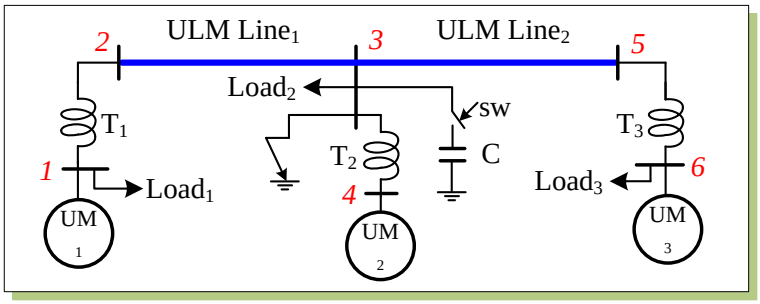
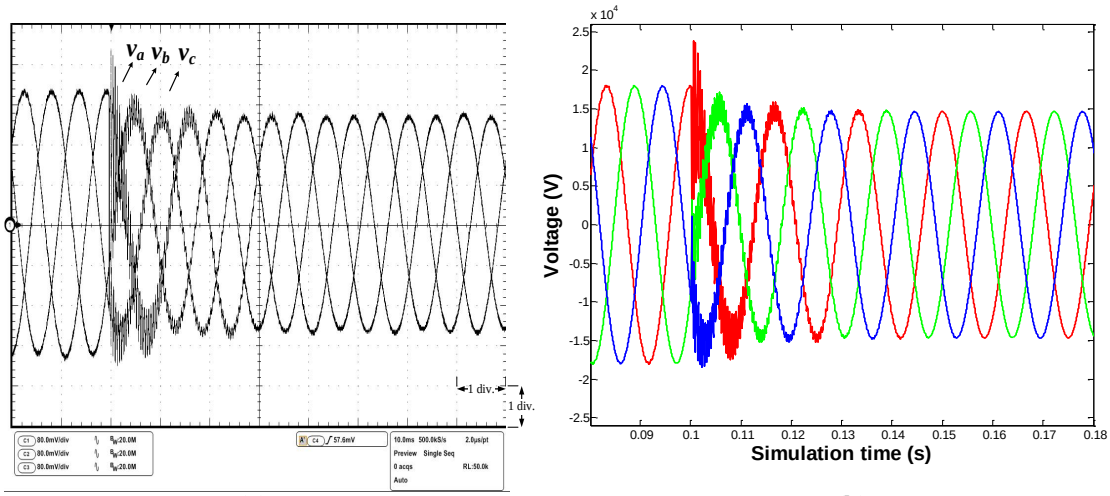


Figure 5.21: Single-line diagram of power system for the Case Study.

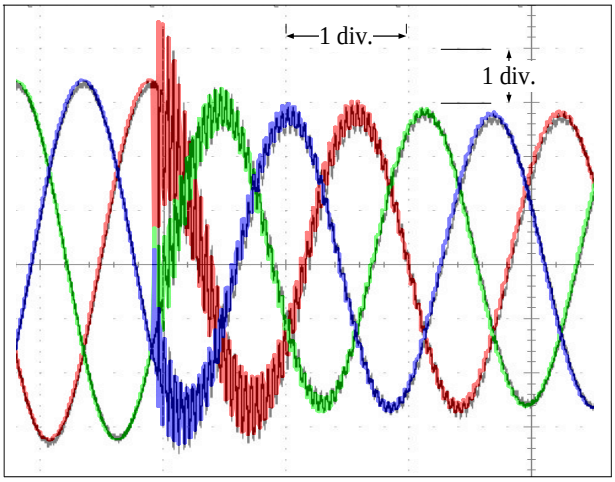
is almost 90% full on logic elements and 70% full on the multipliers. Depending on the size of the system and the time-step required for a certain transient, more UM and ULM modules can be replicated on a larger FPGA to either reduce the time-step or accommodate a large system.

5.6 Summary

This chapter describes a digital hardware emulation of UM and ULM models for real-time electromagnetic transient simulation on the FPGA. Taking advantages of inherent parallel architecture of FPGA, the hardware is paralleled and fully pipelined to achieve efficient real-time simulation of electromagnetic transients. An example system with 3 UM and 2 ULM models is simulated within a 6.72 μs time-step on a 130MHz FPGA clock frequency. The captured real-time waveforms are validated by off-line EMTP-RV simulation results.

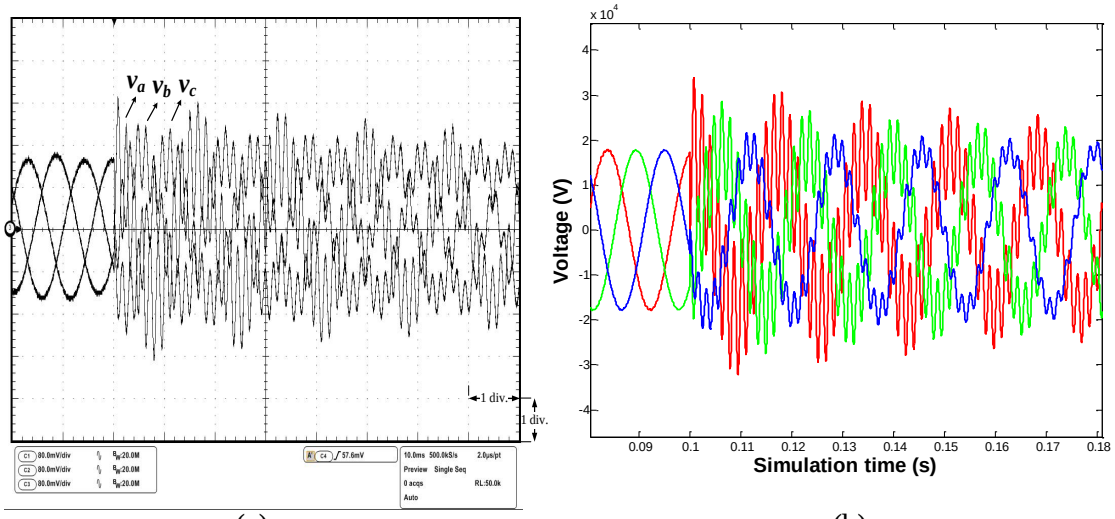


(a) (b)



(c)

Figure 5.22: Real-time oscilloscope traces (a), off-line EMTP-RV simulation (b), and zoomed and superimposed view (c) of the three-phase voltages at Bus 2 during a three-phase fault at Bus 3. Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 5.2kV.



(a) (b)

Figure 5.23: Real-time oscilloscope traces (a) and off-line EMTP-RV simulation (b) of the three-phase voltages at Bus 3 during a capacitor switching at Bus 3. Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 10.4kV.

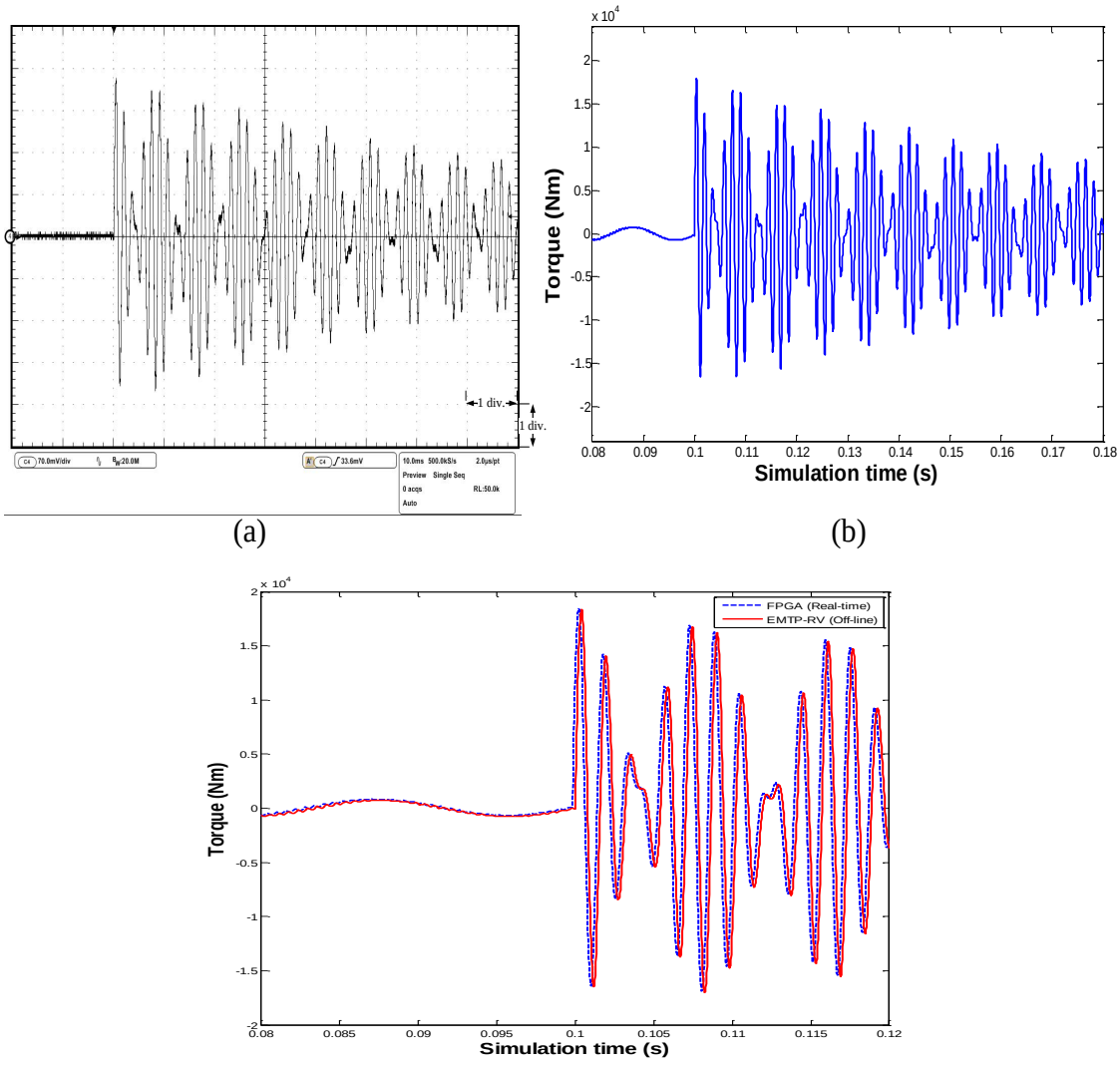


Figure 5.24: Real-time oscilloscope traces (a), off-line EMTP-RV simulation (b), and zoomed and superimposed view (c) of the electromagnetic torque of UM_2 during a capacitor switching at Bus 3. Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 4.7kNm.

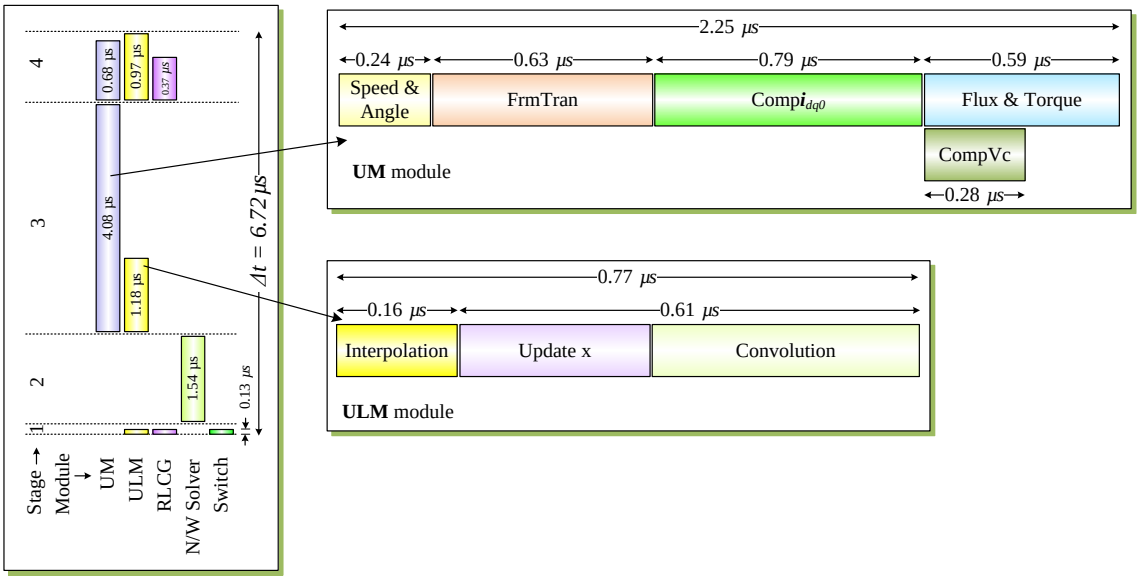


Figure 5.25: Break down of one time-step (Δt) in μs for various stages and modules for the Case Study.

6

Multi-FPGA Hardware Design for Large-Scale Real-Time EMT Simulation

In this chapter[1] a multi-FPGA hardware design for detailed real-time EMT simulation of large-scale power systems is described. First the functional decomposition method is introduced for the hardware emulation of large-scale power system. Then two case studies are provided to show the details of multi-FPGA based hardware design using functional decomposition method for real-time EMT simulation. The real-time simulation results are validated using the off-line simulation software EMTP-RV. The performance and scalability analysis for the proposed multi-FPGA hardware design are also given.

6.1 Introduction

Nowadays the real-time EMT simulators are required not only to reproduce detailed high-frequency transients but also to be able to simulate large-scale power system transients due to the increasing complexity of modern power systems. But in practical large-scale real-time EMT simulators accuracy and computational efficiency are conflicting requirements that have ramifications in terms of simulator hardware and cost. A realistic reproduction of high-frequency transients requires detailed modeling of power system components and a small simulation time-step. Thus, large system size entails excessive computational burden. To lower computational burden using simplified modeling or a large time-step would lower fidelity of the simulation. Inevitably, to meet real-time constraints and to accommodate large system sizes, a compromise is needed in component model complexity

[1]Material from this chapter has been submitted: Y. Chen and V. Dinavahi, "Multi-FPGA digital hardware design for large-scale real-time EMT simulation of power systems", *IEEE Trans. on Industrial Electronics*, pp. 1-8, 2011.

to curtail simulator cost at the expense of accuracy. In such cases simplified modeling of system components is tolerated. Simplified models include PI or traveling wave representation for lines and cables, Thévenin equivalent representation for machines, and switched piecewise linear approximation for nonlinear elements.

6.2 Functional Decomposition Method for Large-Scale Real-Time EMT Simulation

Parallel processing is extensively used in existing real-time simulators to execute large-scale system models cooperatively on multiple sequential processors. This parallel processing relies on the fundamental premise that a power system can be decomposed into smaller subsystems (Fig. 1.3) due to the natural travel time delay of the transmission line or cable which provides the decoupling necessary for the subsystem calculations to occur without timing conflict; thus one or more subsystems can be assigned to multiple sequential processors which share the computational workload, subject to the condition that the simulation time-step is less than the travel time on the link lines. However, this method has some drawbacks:

1. If there is no real transmission line or cable connecting any two subsystems or if two neighboring subsystems are tightly coupled, fictitious lines or cables with travel time are necessary to partition the network. Such artificial lines introduce errors in the frequency response of the simulated transient; although the errors can be compensated or minimized, the location and the length of the link lines still need to be carefully chosen to maintain accuracy.

2. After executing their respective calculation the sequential processors need to exchange subsystem data with each other at the cost of extra communication latency within the time-step which limits the achievable bandwidth of the real-time simulator.

3. The partitioning scheme to divide the original system into smaller subsystems is arbitrary at best, usually based on the experience and the specific requirements of the user, albeit some simulators such as HYPERSIM have an automatic partitioning method.

4. Based on a given network topology it is quite possible to find a large subsystem which cannot be decomposed further leading to uneven computational workload for the processors. In such cases, the overall computational bandwidth of the real-time simulator is limited by the speed of the processor to which the largest subsystem is assigned.

The traditional method for accommodating large network sizes on limited simulator hardware is to use a frequency-dependent network equivalent (FDNE) where the original

system is divided into a study zone (modeled in detail) and an external system (modeled as lumped RLCG groups derived from approximating the frequency response of the external system at the interfacing port), as shown in Fig. 1.4. While network equivalents are still used in some off-line and real-time simulators, the prevailing requirement is to represent the original system in full detail without using simplified equivalents.

Digital hardware emulation of large-scale networks has the potential to alleviate all of the above compromises and uncertainties inherent in large-scale real-time EMT simulation, using the FPGA as the core simulator hardware. A *functional decomposition* method is introduced for allocating the system model calculation to the multi-FPGA hardware resource. This method of power system decomposition relies on systematically clustering the model calculation of system components on individual FPGAs based on the component functionality independent of the network topology. As seen in Fig. 6.1, a generic power system composed of lines, generators, nonlinear elements, loads, buses, circuit breakers, etc., can be mapped into a multi-FPGA architecture such that each processing hardware (PH) is responsible for the model calculation of one type of system component. For example, all transmission lines are simulated in the specified PH. Conventional network partitioning using transmission line links is no longer required to simulate a large-scale system. Alternately, the models for each type of system component are calculated simultaneously in their own PHs. As shown later, functional decomposition lends itself to the full exploitation of pipelining and hardware parallelism inherent in an FPGA. It does not require artificial lines or cables of fixed latency to be inserted, rendering the hardware emulation true to the original system. It removes the uncertainty related to partition boundaries, while allowing detailed modeling of all system components with an even distribution of computational workload. Since system components of the same type are clustered together, this method naturally allows multi-rate simulation where multiple time-steps can be chosen for various functional types to account for model complexity or to increase computational efficiency.

6.3 Functional Module and Parallelism

The hardware modules corresponding to the system functional components are implemented in the $65nm$ Stratix III EP3SL340 FPGA from Altera whose logic resources are shown in Table 2.1. The emulated modules include the `RLCG` module for lumped RLCG elements and transformers, the `ULM` module for transmission lines and cables, the `UM` module for machines, the `NR` module for nonlinear elements, the `Switch` module for circuit breakers, and the `Network Solver` module is realized solving the network equation. In addition, a `Control` module is implemented to coordinate the calculations all the other modules. Since each module is hardware independent with respect to the other, the parallelism between various functional modules is achieved naturally. This means that the processing in all modules can be carried out simultaneously. This is *system-level* parallelism

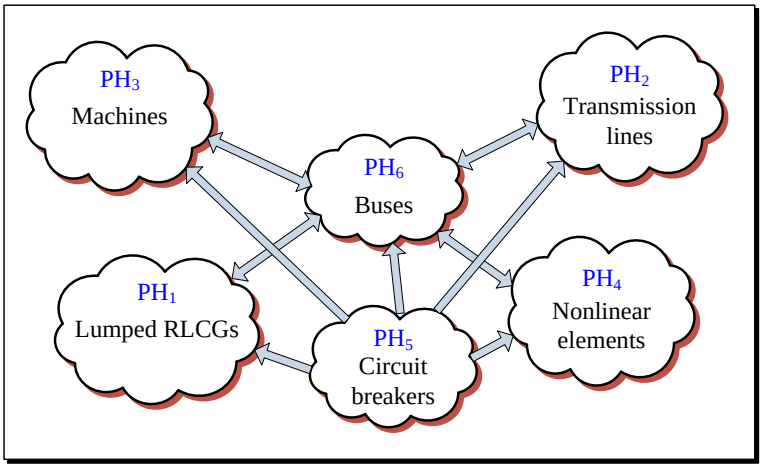


Figure 6.1: Functional decomposition of a power system for hardware emulation.

existing in the FPGA-based real-time EMT emulator. Parallelism also exists within each of the functional modules which have been shown in previous chapters. For example, in the `ULM` module [71], the calculations at both sending-end and receiving-end is conducted fully in parallel. For multi-phase transmission lines the calculation in each phase is also executed simultaneously. This is *module-level* parallelism.

Table 6.1 lists the main logic resources utilized by each functional module in the design. As can be seen the `ULM` module utilizes the most logic resources. To emulate a large-scale power system network in real time, hardware parallelism and pipelining need to be exploited on a larger-scale which can be realized by using multiple hardware functional modules; however, as seen from Tables 2.1 and 6.1, this is hard to achieve on a single FPGA due to the limitations of its logic resources. Multiple FPGAs have to be employed invariably to enable massive parallelism and pipelining for large-scale systems.

Table 6.1: FPGA resources utilized by individual system functional modules

Functional module	Combinational ALUTs	DSP 18-bit multipliers	Memory (Kbits)
`RLCG`	3,701 (1.36%)	8 (1.39%)	123.16 (0.76%)
`ULM`	44,610 (16.49%)	96 (16.67%)	707.27 (4.35%)
`UM`	24,679 (9.12%)	88 (15.27%)	870.74 (5.35%)
`NR`	11,729 (4.33%)	16 (2.78%)	217.39 (1.33%)
`Switch`	124 (0.05%)	0 (0%)	0 (0%)
`N/W Solver`	2,151 (0.79%)	4 (0.69%)	43.8 (0.27%)
`Control`	200(0.07%)	0 (0%)	0 (0%)

6.4 Multiple FPGA Based Hardware Design for Real-Time EMT Simulation

Based on the hardware resource utilization of the individual functional modules, the number of modules of a specific type to replicate and the number of pipelined components per module are decided in the multi-FPGA design. From the EMT user viewpoint there are 3 main variables for real-time simulation: (1) the simulation time-step Δt, (2) number of nodes in the modeled system, and (3) the number of FPGAs employed. These variables influence specific aspects of the real-time EMT simulation: the time-step Δt influences the maximum frequency of the simulated transient; the number of nodes influences the size or scale of the system simulated; and the number of FPGAs influences the hardware cost. Accordingly, there are 3 questions a hardware designer is faced with:

1. For a given system size and hardware configuration, what is the minimum time-step Δt_{min} achievable for real-time EMT simulation?

2. For a given system size and a specified time-step Δt, what is the minimum number of FPGAs required for real-time simulation?

3. For a fixed hardware configuration and a specified time-step Δt, what is the largest power system network that can be simulated in real time?

The hardware designs presented in this section attempt to reveal the answers to these questions.

The DN7020k10 multi-FPGA board from The Dini Group® is utilized to realize these designs. This board is populated with 10 Altera Stratix III EP3SL340 FPGAs arranged in a 2×5 matrix as shown in Fig. 6.2. Taken together $FPGA_0$-$FPGA_9$ provide 3,380,000 equivalent logic elements, 162,720 memory Kbits, and 5,760 18-bit multipliers. Ample pin connections are also provided between adjacent FPGAs; each pair of adjacent FPGAs share a maximum of 220 pins for high speed bidirectional data transfer. A 48-pin bus (`MainBus`) is connected to all FPGAs. The FPGA interconnects are either single-ended or LVDS. The multi-FPGA board has programmable clock synthesizers (2KHz-710MHz), and global clock networks that reach every FPGA. The hardware designs are performed in VHDL on the host PC using the Altera Quartus II environment. The configuration files (bitstreams) are downloaded from the host PC into the individual FPGAs using the `JTAG` interface through a USB cable. Fast configuration using Compact FLASH is also possible when the debugging is finished. A 125 MSPS DAC card is connected through a high-speed QSE connector to the multi-FPGA board to enable the capture of real-time results on the oscilloscope.

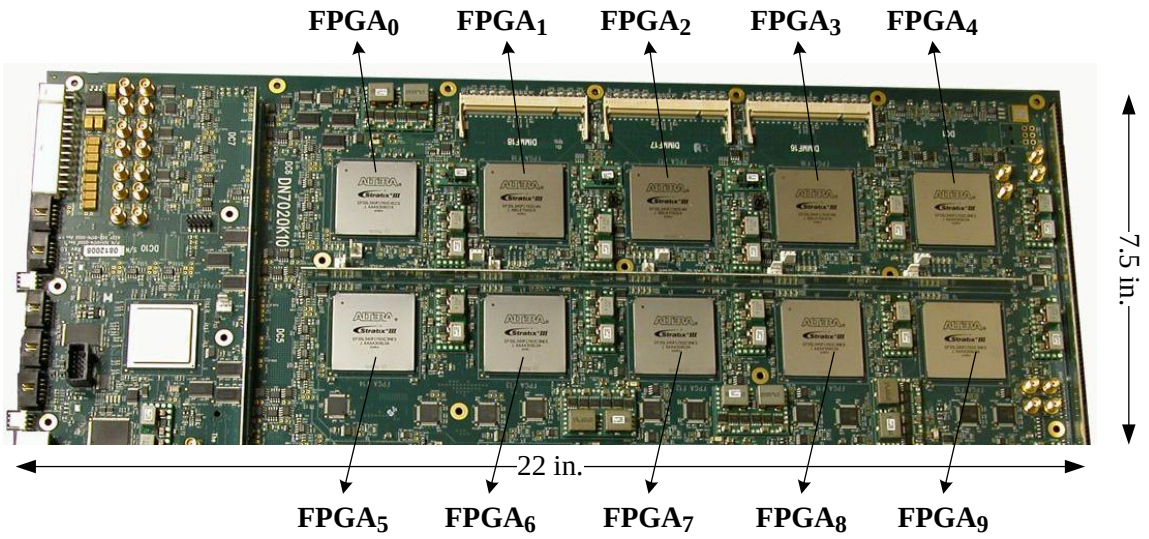


Figure 6.2: Multi-FPGA prototyping board.

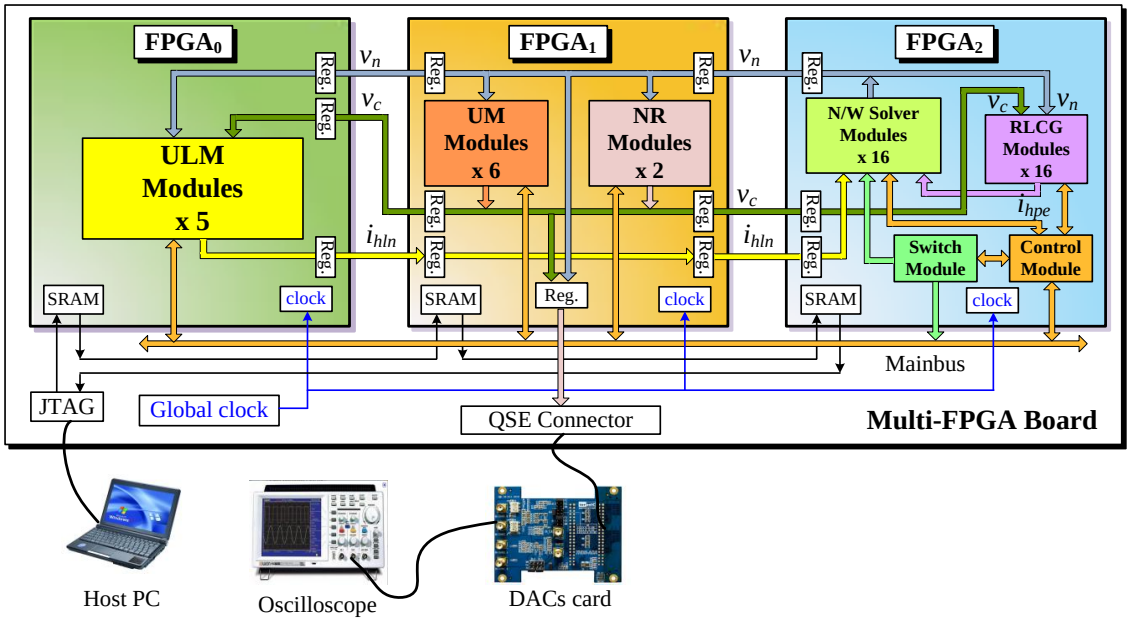


Figure 6.3: 3-FPGA hardware architecture for real-time EMT simulation in Case Study I.

6.4.1 Case Study I: 3-FPGA Hardware Design

This case study shows the design details of a 3-FPGA functionally decomposed real-time EMT simulator. As can be seen in Fig. 6.3, $FPGA_0$ is fully employed to realize 5 `ULM` hardware modules; $FPGA_1$ is used to implement 6 `UM` and 2 `NR` modules, whereas $FPGA_2$ implements other modules which includes 16 `Network Solver`, 8 `RLCG`, 1 `Switch`, and 1 `Control` modules. This arrangement of the various modules into the 3 FPGAs minimizes the interconnected signals between FPGAs. This is important because on the one

hand, the limited FPGA pins cannot support massively parallel data input/output, while on the other hand, on a multi-FPGA board only neighboring FPGAs are usually interconnected and support high-speed data transfer, for example, the 225MHz data transfer for single-ended signals on adjacent FPGAs on the DN2070k10 board. In the 3-FPGA hardware configuration the transferred signals include the node voltages $\boldsymbol{v}_n$ which are calculated in the `Network Solver` modules in $FPGA_2$ and are sent to $FPGA_1$ and $FPGA_0$; the compensated voltages $\boldsymbol{v}_c$ which are calculated in the `UM` and `NR` modules in $FPGA_1$ and are sent to $FPGA_0$ and $FPGA_2$; and the line history currents $\boldsymbol{i}_{hln}$ which are calculated in the `ULM` modules in $FPGA_0$ and are sent to $FPGA_2$ via $FPGA_1$. Buffer registers are inserted for signal input and output between the 3 FPGAs. The `MainBus` is used by the `Control` module in $FPGA_2$ to send control signals and receive acknowledge signals. The `Switch` module in $FPGA_2$ also uses it to send switch status signals. The logic resource utilization for this 3-FPGA design is shown in Table 6.2. As can be seen the hardware space for $FPGA_0$ and $FPGA_1$ are filled to capacity with modules of the specific functional components with little room for further expansion, while $FPGA_2$ still has leftover capacity. Although some resources may appear to be under-utilized, the resource that reaches its limit first determines the maximum capacity for any specific FPGA, e.g. DSP block in the $FPGA_1$. Table 6.2 also shows the f_{max} of each FPGA. The f_{max} is the maximum clock frequency that the designed hardware can operate at, and it is calculated based on the longest signal path latency in the design. It is obvious that the higher the resource utilization of the FPGA, the lower is the f_{max}, i.e., $f_{max0} < f_{max1} < f_{max2}$. Furthermore, although multi-rate simulation is clearly available in various different modules, for simplicity a 100MHz clock frequency is chosen for all FPGAs in this design.

To test the 3-FPGA hardware design a sample power system shown in Fig. 6.4 is modeled. It is a modified version of the IEEE 39-bus New England test system. This system consists of 35 three-phase transmission lines modeled using the ULM; 10 three-phase generators modeled using the UM model; 19 three-phase loads modeled using RLCG elements, 11 three-phase transformers modeled using equivalent RLCG elements, and 3 series compensation capacitors, resulting in a total of 99 lumped RLCG elements; and 3 three-

Table 6.2: Resource utilization for the 3-FPGA hardware design

	$FPGA_0$	$FPGA_1$	$FPGA_2$
Logic utilization	96%	80%	35%
-Combinational ALUTs	84%	64 %	27%
-Dedicated logic registers	44%	37 %	14%
DSP block 18-bit elements	83%	97%	22%
Block memory bits	22%	35%	17%
PLLs	8%	8%	8%
Pins	13%	32%	52%
f_{max} (MHz)	100.4	136.9	170.8

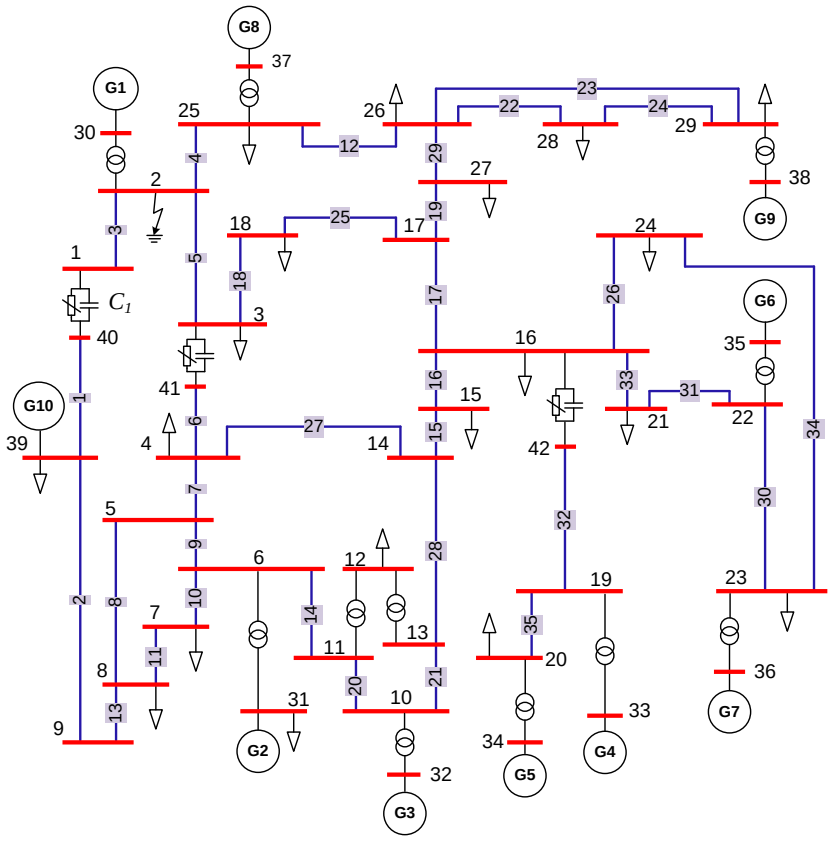


Figure 6.4: Single-line diagram of the power system modeled in Case Study I.

phase nonlinear surge arresters which protect 3 series compensation capacitors. The total number of nodes in the network is 126.

The spatio-temporal design workflow to model this system in real time on the 3-FPGA hardware design is shown in Fig. 6.5. As can be seen, the 35 lines are allocated to the 5 `ULM` modules with 7 lines pipelined through each module. The 10 generators are allocated to the 5 `UM` modules with 2 generators pipelined per module and with one module empty. The 99 RLCG elements are allocated into 8 `RLCG` modules (12 RLCGs each for the first 5 modules and 13 RLCGs each for the remaining 3 modules). The 3 surge arresters are allocated to the 2 `NR` modules (2 in the first module and 1 in the second). The 126 node voltages are solved in 16 `Network Solver` modules (8 nodes each in first 14 modules and 7 nodes each in the other 2 modules). Fig. 6.5 also shows the constitution of a simulation time-step in the 3-FPGA real-time EMT simulator. One simulation time-step Δt has 3 periods: *Period I* in which the node voltages $\boldsymbol{v}_n$ (without taking into account the electric machines and nonlinear elements) are solved in `Network Solver` module; *Period II* in which the `UM` and `NR` modules start to solve their respective equations and the compensated voltages $\boldsymbol{v}_c$ are calculated; meanwhile, the `ULM` modules compute part of their convolutions; and *Period III* in which the `RLCG`, `UM`, and `ULM` modules update their history terms. The minimum

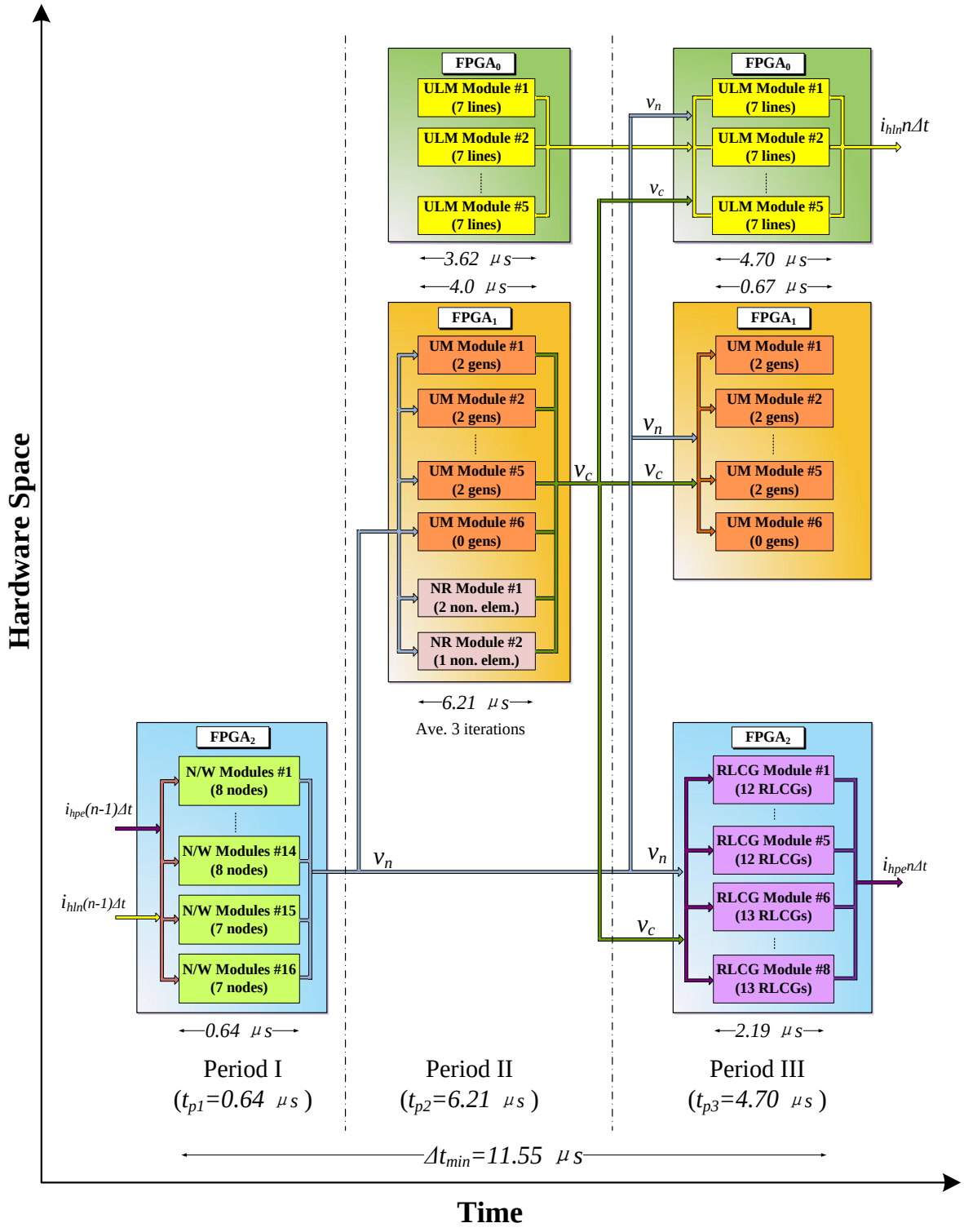


Figure 6.5: Spatio-temporal design workflow for the 3-FPGA real-time EMT simulator for Case Study I.

time-step Δt achieved for modeling the system in Fig. 6.4 in the 3-FPGA hardware design is 11.55 μs, with *Period I* taking the minimum execution time of $0.64\mu s$ for the `Network Solver` modules, and *Period II* taking the maximum execution time of $6.21\mu s$ for the `NR` modules for an average 3 iterations for the nonlinear solution. In *Period II*, the execution time is determined by the `NR` module in FPGA_1, while in *Period III* the execution time is determined by the `ULM` module in FPGA_0. As a comparison, the execution time per simulation step of this system modeled in EMTP-RV with $\Delta t = 12\mu s$ running on a PC (AMD Phenom II 955 CPU, 3.2GHz, 4 cores, 16GB system memory) is $192\mu s$, as shown in Table 6.3.

A three-phase-to-ground fault at Bus 2 which occurs at $t = 0.05s$ is emulated in the 3-FPGA hardware design. Fig. 6.6 (a) shows the three-phase voltages waveforms at Bus 1 captured from a real-time oscilloscope connected to the DAC card. The voltage transient lasts about 2 cycles and its peak value falls from 20kV to 15kV. Identical behavior can be observed from Fig. 6.6 (b) which shows the off-line EMTP-RV simulation and (c) which shows the zoomed and superimposed view.

6.4.2 Case Study II: 10-FPGA Hardware Design

This case study utilized all the 10 FPGAs to exploit maximum hardware space on the multi-FPGA board. The 10-FPGA hardware architecture is shown in Fig. 6.7. FPGA_0, FPGA_1, and FPGA_2 remain the same as in Case Study I. FPGA_3 is employed for realizing the `UM` and `NR` modules as FPGA_1. FPGA_4 through FPGA_9 are employed for realizing `ULM` modules. In Fig. 6.7, $\boldsymbol{v}_{c1}$, $\boldsymbol{v}_{c3}$ denote the compensated voltages computed in FPGA_1 and FPGA_3, respectively, and $\boldsymbol{i}_{Fk}$ $\{k = 0, 4, 5, 6, 7, 8, 9\}$ are history current vectors calculated in FPGA_k.

The number of any specific functional modules implemented in this multi-FPGA architecture depends on three criteria: demand, logic utilization, and module execution time. The demand is determined by the number of each type of functional component in the power system. For example, there are many more transmission lines than generators in a power system; thus the number of the `ULM` modules is expected to be more than that of the `UM` modules. The logic utilization of modules (Table 6.1) determined the possible number of modules implemented on any FPGA. The module execution time affects the time-step directly. More parallel modules are required for those components that have longer execution times. The detailed timing analysis for each functional module is discussed in the next section.

A large-scale power system shown in Fig. 6.8 is used to test the 10-FPGA real-time EMT simulator. This system is constructed by replicating the Case Study I system 10 times and interconnecting using transmission lines. The augmented system consists of:

- 376 three-phase transmission lines modeled using the ULM.

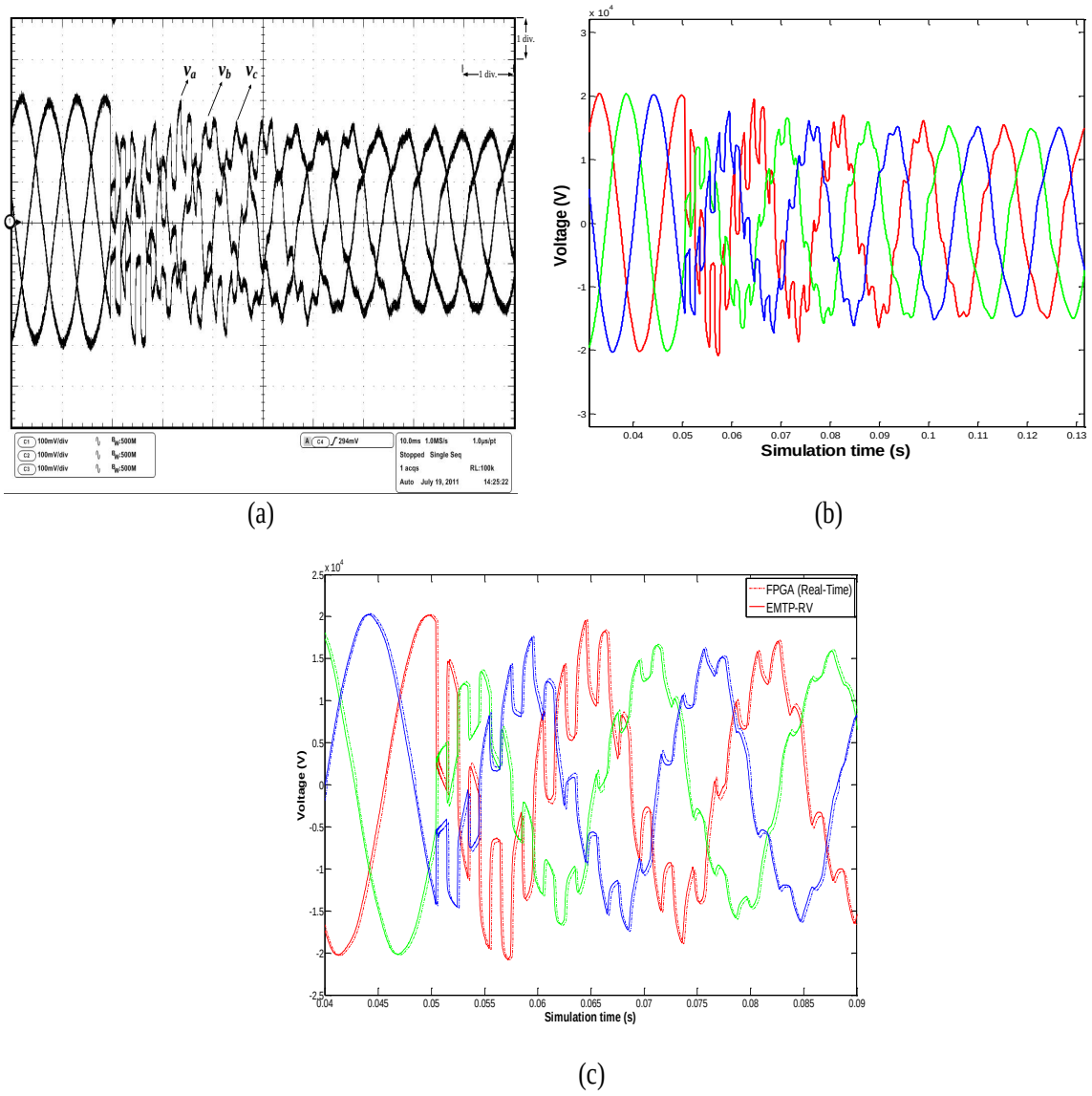


Figure 6.6: Real-time oscilloscope traces (a), off-line simulation results from EMTP-RV (b), and zoomed and superimposed view (c) of Bus 1 voltages for a three-phase fault at Bus 2, Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 6.8kV.

- 100 three-phase generators modeled using the UM model.
- 190 three-phase loads modeled using RLCG elements.
- 110 three-phase transformers modeled using equivalent RLCG elements with a total 990 lumped RLCG elements.
- 30 three-phase nonlinear surge arresters which protect 30 series compensation capacitors.

The number of network nodes in this system is 1260.

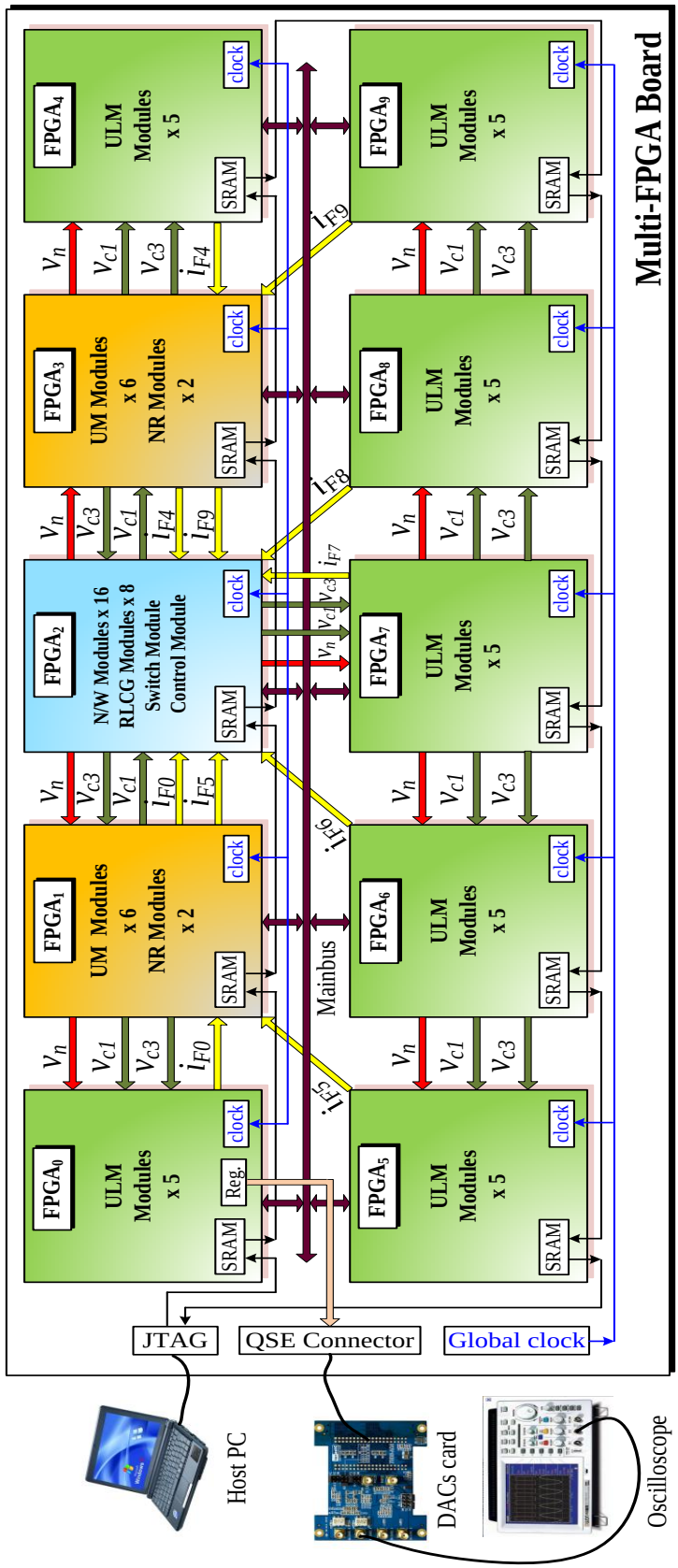


Figure 6.7: 10-FPGA hardware architecture for real-time EMT simulation in Case Study II.

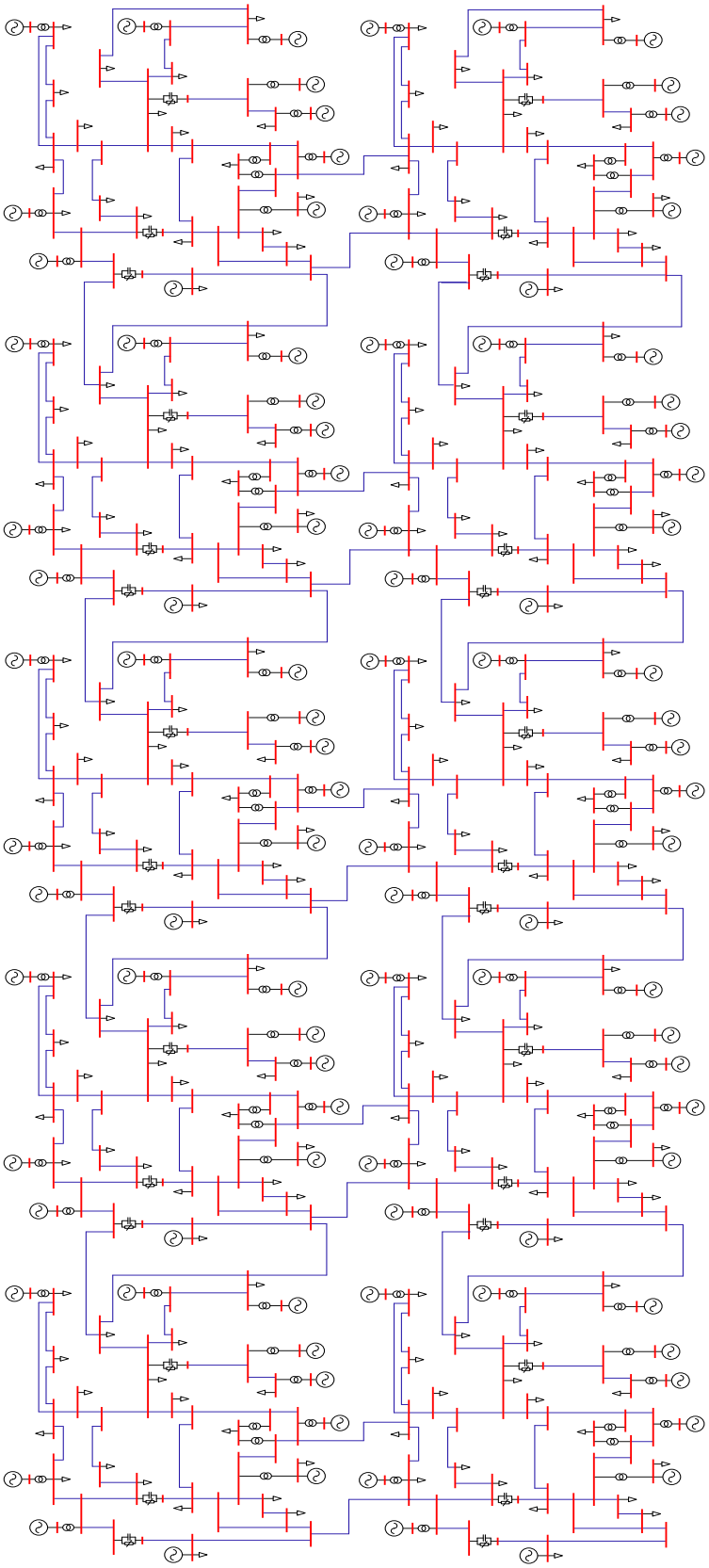


Figure 6.8: Single-line diagram of the power system modeled in Case Study II.

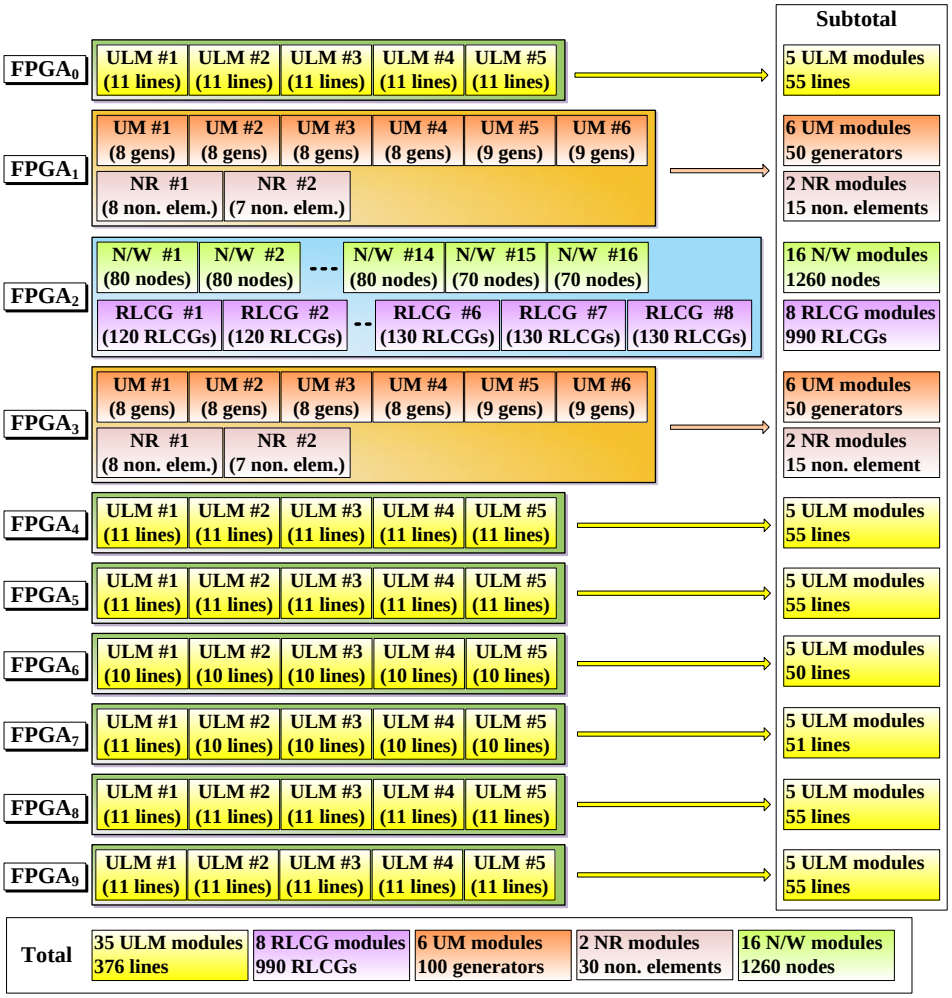


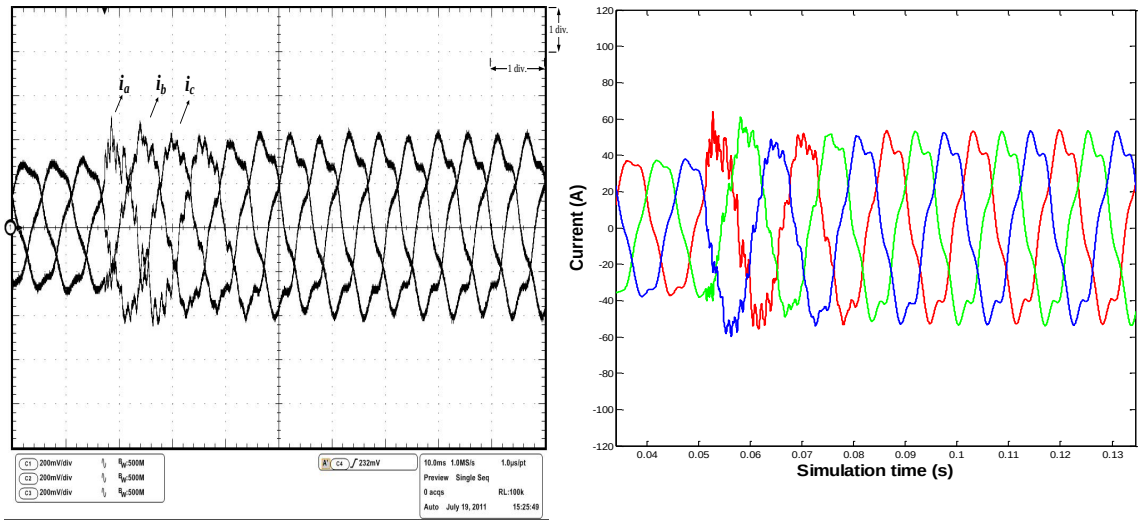
Figure 6.9: Allocation of components of Case Study II in the 10-FPGA design.

The allocation of all components into this 10-FPGA design is shown in Fig. 6.9. The overall clock frequency driving all FPGAs for the 10-FPGA hardware design is 100 MHz. The achieved minimum time-step Δt in the 10-FPGA hardware design is 36.12 μs. As shown in Table 6.3, the execution time per simulation step of EMTP-RV is $2120\mu s$.

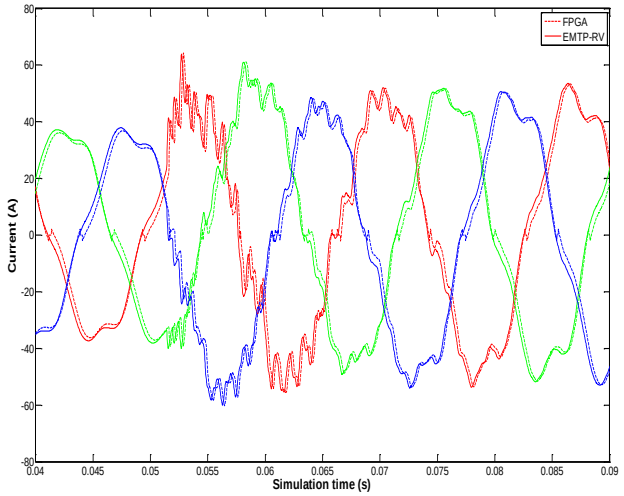
Table 6.3: EMTP-RV execution time for the two case studies

	Δt	Execution time per simulation step
Case Study I	12 μs	192 μs
Case Study II	$36\mu s$	2120 μs

Fig. 6.10 (a) shows the terminal currents of generator $G9$ during the three-phase-to-ground fault at Bus 2, where the transients can be seen clearly. Again, the detailed agreement between real-time and off-line simulations can be observed in Fig. 6.10 (b) and (c).



(a) (b)



(c)

Figure 6.10: Real-time oscilloscope traces (a), off-line simulation results from EMTP-RV, and zoomed and superimposed view (c) of generator $G9$ terminal currents during a three-phase fault at Bus 2, Scale: x-axis: 1div. = 10ms, y-axis: 1div. = 25A.

6.5 Performance and Scalability of the Multi-FPGA Real-Time Hardware Emulator

To answer the 3 questions posed earlier, first a detailed analysis of the time-step is in order. As shown in Figs. 6.5 and 6.9, the simulation time-step Δt can be determined as

$$\Delta t = t_{p1} + t_{p2} + t_{p3}, \tag{6.1}$$

where t_{p1}, t_{p2}, and t_{p3} are the execution times in *Period I*, *Period II*, and *Period III*, respectively, which are calculated as

$$t_{p1} = t_{NW}, \tag{6.2a}$$

$$t_{p2} = max\{t_{ULM}, t_{UM}, t_{NR}\}_{p2}, \tag{6.2b}$$

$$t_{p3} = max\{t_{ULM}, t_{UM}, t_{RLCG}\}_{p3}, \tag{6.2c}$$

where t_{NW}, t_{ULM}, t_{UM}, t_{NR}, t_{RLCG} are execution times for `Network Solver` module, `ULM` module, `UM` module, `NR` module, and `RLCG` module, respectively. The execution time for each module is calculated according to the total hardware latency of the module and pipelined components fed into the module. For example, t_{ULM} in *Period II* can be determined as

$$t_{ULM} = (34 + 8 * N_{line}) * T_f, \tag{6.3}$$

where the number '34' denotes the total hardware latency in terms of clock cycles; N_{line} is the number of lines pipelined into the module, and T_f is the working clock period of FPGA (10 ns in this design). The number '8' is related to the assumption that each transmission line has a 9th-order ($N_p = 9$ in (5.24)) fitted rational functions for the characteristic admittance matrix and total 13th-order ($N_{p,1} = 4$ and $N_{p,2} = 9$ in (5.25)) of fitted rational functions in 2 ($N_g = 2$ in (5.25)) propagation modes for the propagation function matrix. The execution time of the `ULM` module in *Period II* and *Period III* with respect to the number of lines pipelined through the module is plotted in Fig. 6.11 (b). Similarly, Fig. 6.11 (a) shows t_{RLCG} with respect to the number of RLCG elements; Fig. 6.11 (c) shows t_{UM} with respect to the number of machines; Fig. 6.11 (d) shows t_{NR} with respect to the number of nonlinear elements; and Fig. 6.11 (e) shows t_{NW} with respect to the number of network nodes.

The performance of the proposed multi-FPGA hardware design is investigated by the minimum time-step with respect to the simulated system size. The Case Study I test system (Fig. 6.4) is used as a reference system to determine the number of RLCG elements, transmission lines, machines, and nonlinear elements with respect to the system size quantified in terms of the number of network nodes. Based on this information, the minimum time-step is calculated for a given size of power system, as shown in Fig. 6.12. As can be seen in this figure, the minimum time-step of 11.55μs and 36.12μs are achieved for Case Study I for a system of 126 nodes in the 3-FPGA design and Case Study II for a system of 1260 nodes in the 10-FPGA design, respectively. Fig. 6.12 also shows the largest system size that can be simulated with a specified time-step. For example, with a 50μs time-step using detailed modeling a system of 850 nodes can be simulated on the 3-FPGA design, while a system of 1860 nodes can be simulated on the 10-FPGA hardware design. Note that with simplified models much larger systems can be simulated with the same time-step.

The number of FPGAs required to carry out real-time EMT simulation varies with the specific time-step and the given system size. Fig. 6.13 shows the surface plot of the variation of number of FPGAs required with respect to system size, and the minimum time-step

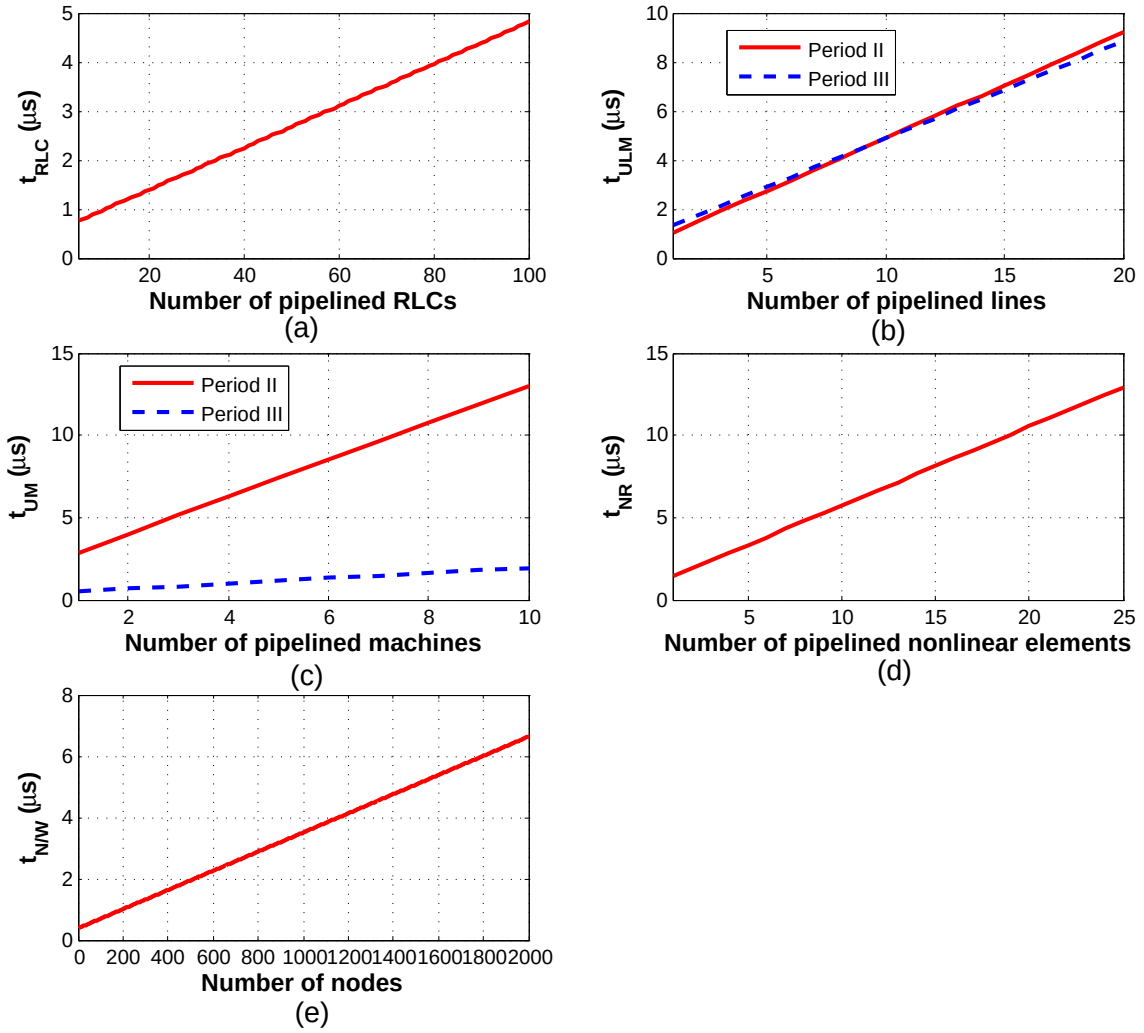


Figure 6.11: Execution time of each functional module with respect to the number of pipelined elements per module in the multi-FPGA real-time EMT simulator.

achieved. This plot shows the possible combinations of these 3 variables, and the two case studies presented before represent two extremities of this 3-D surface. Clearly, the variation of the minimum time-step is nonlinear with respect to the system size and the number of FPGAs used. The slope of Δt versus system size curve tends to decrease as the number of FPGAs increases. In general, for a fixed large-scale system size employing more FPGAs would allow to achieve a lower time-step for real-time EMT simulation but with diminishing returns. Nevertheless, the achieved time-step would be so small as to allay any concerns related to detailed system modeling. Meanwhile, larger and faster FPGAs would allow more parallel functional modules to be emulated, and would also raise the maximum frequency (f_{max}) of the design, which will ultimately lead to further reduction of time-step.

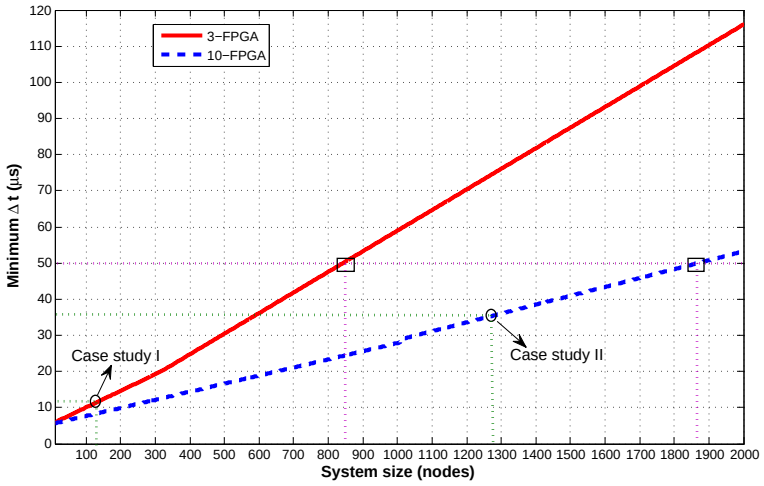


Figure 6.12: Minimum time-step achieved in the 3-FPGA and 10-FPGA hardware designs for real-time EMT simulation.

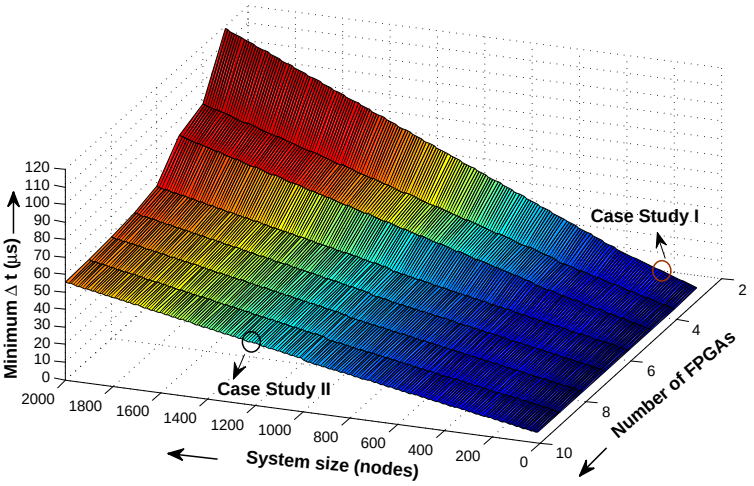


Figure 6.13: Variation of number of FPGAs with system size and the time-step in the multi-FPGA hardware design.

6.6 Summary

Hardware emulation on FPGAs makes it possible to overcome many of the limitations related to the real-time EMT simulation of large-scale power systems by allowing the user to achieve such level of detail that is seldom achieved on traditional sequential processors. This chapter proposed a multi-FPGA design based on the functional decomposition methodology for hardware emulation of systems using detailed modeling of the individual components. The functional decomposition method circumvents the need for artificial lines for dividing the system, reduces subsystem communication latencies, and is independent of a specific partitioning scheme. Furthermore, this method is ideally suited for

achieving full parallelism and pipelining in the implemented system modules on the FPGAs. Although a single time-step is chosen to illustrate this method, multi-rate simulation with multiple time-steps for various functional components is clearly possible. Two case studies involving a three-phase 42-bus and a 420-bus power systems are presented to analyze the performance of the proposed hardware design with detailed modeling of the power system components. The achieved minimum time-steps for these systems are $11.55\mu s$ and $36.12\mu s$ on the 3-FPGA and the 10-FPGA hardware designs, respectively, on a clock frequency of 100MHz. Smaller time-steps and even larger system sizes can be easily achieved as the design is fully scalable and extensible for an FPGA-cluster. Every new generation of FPGA technology features devices with significant increase in logic resource count and clock speed which should satisfy even the most demanding real-time EMT simulation.

7 Conclusions and Future Work

Endless demand of electrical energy is making modern power systems increasingly complex as well as large in size. Highly accurate and efficient modeling and simulation tools are required for analyzing power systems. This leads to the great importance of real-time simulation of electromagnetic transients for planning, design, operation, and control of modern power systems. The EMT simulation of large-scale power systems in real-time using detailed modeling is computationally very demanding and challenging in traditional general purpose CPU and DSP based real-time simulator due to their sequential processed software implementation.

The hardware emulation of complicated model algorithms to achieve high performance computation is drawing more attention nowadays. Due to their inherent parallel architecture, high clock speed, and high logic resource capacity, FPGAs are increasingly being used as core computational hardware in many applications which traditionally used sequential general purpose processors or digital signal processors for carrying out intensive calculations such as electromagnetic transient simulation.

This thesis describes digital hardware emulation of power system component models for large-scale real-time electromagnetic transient simulation. The summary of the completed thesis work and suggestions for future work are presented in this chapter.

7.1 Contributions of this Thesis

The main contributions of this thesis can be summarized as follows:

- The complete real-time EMT simulator is realized in the FPGA. The emulated power system components include linear lumped RLCG elements, frequency-dependent transmission lines (FDLM), supply sources, and circuit breakers. The network is

solved efficiently on a proposed fast network solver hardware module exploiting sparse matrix techniques. This is the first time that real-time EMT simulation using detailed models is carried out in a single FPGA. Through this work FPGA shows great potential on accurate and efficient hardware emulation of EMT.

- A novel parallel EMT solution algorithm is proposed. The potential parallel processing in the EMT solution is investigated in order to accommodate the parallel architecture of the FPGA. Moreover, parallel processing inside each individual component model is realized as well. For example, the independent hardware units are developed for emulation of both ends and each phase of transmission lines.

- An iterative Newton-Raphson nonlinear solver is implemented in the FPGA. Real-time simulation of nonlinear elements is always challenge due to the iterative nature of the solution algorithm on the sequential hardware like CPU and DSP. Taking advantages of compensation method, proposed fast sparse matrix processing hardware module, and proposed parallel Gauss-Jordan elimination hardware module, the nonlinear elements such as nonlinear surge arresters and nonlinear inductances are able to be emulated accurately and efficiently in the FPGA.

- Accurate EMT simulation of power systems requires detailed and generalized models of rotating machines and transmission lines. The universal machine (UM) model is a sufficiently generalized machine model which can accurately represent several types of rotating machines for transient studies. The universal line model (ULM) is the most accurate and general model for both overhead line and underground cable in both symmetrical and unsymmetrical scenarios. These two models have been implemented in off-line simulation programs successfully; however, due to the complexity of UM and ULM models and limited computational power of traditional sequential hardware, the real-time implementation of UM and ULM models is challenging. It is the first time that these two models are emulated in the FPGA for real-time EMT simulation. The emulated hardware modules are fully paralleled and deeply pipelined.

- For real-time EMT simulation of large-scale power systems, multiple FPGAs are necessary. A novel functional decomposition method is introduced to allocate power system components into the corresponding hardware functional modules in the multiple FPGA architecture. A 10-FPGA hardware platform is used to provide proof-of-concept. A large-scale power system with 1260 nodes is simulated in real-time. This is the first time that real-time EMT simulation of large-scale power systems is realized on a multi-FPGA architecture.

- An IEEE 32-bit floating-point number representation is used for high accuracy throughout the EMT simulation. All hardware arithmetic units designed are deeply pipelined

to achieve highest computation throughput. The whole design is based on VHDL language for fast portability and extensibility on any FPGA platform.

7.2 Directions for Future Work

The following topics are proposed for future work:

- More complex geometrical models of electrical machines such as permeance network model (PNM) could be emulated in the FPGA. PNM, also known as reluctance network models or magnetic equivalent circuits, is considered as a compromise between finite element (FE) analysis and lumped parameter d-q models. Its advantages are the relatively low computing time compared to FE and the high accuracy, achieved through a division of the geometry. Using PNM fault conditions such as stator winding inter-turn short-circuit, broken rotor bars and end rings, and air-gap eccentricity in induction machines can be studied in real-time.

- Sophisticated power electronic apparatus and their digital control systems are finding increasing applications in electric power systems at generation, transmission, distribution, and utilization levels. They constitute a higher computational burden (due to higher number of switches) and require smaller time-steps (due to high switching frequency and the need for accounting the switching signals accurately), thus FPGAs have been widely employed. In EMTP-type off-line simulation softwares, power electronic apparatus are modeled in system-level which it is not able to reproduce the device nonlinear characteristics realistically. Using FPGA, the power electronic can be modeled in device-level allowing investigating the switching transients, power losses, and thermal characteristics of the device.

- The transformers are modeled by Thévenin equivalent network in the FPGA-based real-time EMT simulator in this thesis. Accurate modeling of transformer taking into account the nonlinear phenomena such as hysteresis and the topology of the transformers for real-time EMT simulation would be a future area of interest.

Bibliography

[1] H. W. Dommel and W. S. Meyer, "Computation of electromagnetic transients", *in Proc. IEEE,* vol. 62, no. 7, pp. 983-993, April 1974.

[2] H. W. Dommel, "Techniques for analyzing electromagnetic transients", *IEEE Comput. Appl. Power,* vol. 10, Issue 3, pp. 18-21, July 1997.

[3] N. Watson and J. Arillate, *Power systems electromagnetic transients simulation,* The Institution of Engineering and Technology, London, U. K.,2003.

[4] H. W. Dommel, "Digital computer solution of electromagnetic transients in single and multiphase networks", *IEEE Trans. on Power Apparatus and Systems,* vol. PAS-88, no. 4, pp. 388-399, April 1969.

[5] H. W. Dommel, *EMTP theory book,* Bonneville Power Administration, 1984.

[6] W. Ren, M. Sloderbeck, M. Steurer, V. Dinavahi, T. Noda, S. Filizadeh, A. R. Chevrefils, M. Matar, R. Iravani, C. Dufour, J. Belanger, M. O. Faruque, K. Strunz, and J. A. Martinez, "Interfacing issues in real-time digital simulators", *IEEE Trans. on Power Delivery,* vol. 26, no. 2, pp. 1221-1230, April 2011.

[7] R. M. Mathur and X. Wang, , "Real-time digital simulator of the electromagnetic transients of transmission lines", *IEEE Trans. on Power Delivery,* vol. 4, no. 2, pp. 1275-1280, April 1989.

[8] X. Wang and R. M. Mathur, "Real-time digital simulator of the electromagnetic transients of transmission lines with frequency dependence", *IEEE Trans. on Power Delivery,* vol. 4, no. 4, pp. 2249-2255, October 1989.

[9] M. Kezunovic, J. K. Bladow, and D. M. Hamai, "Transients computation for relay testing in real-time", *IEEE Trans. on Power Delivery,* vol. 9, no. 3, pp. 1298-1307, July 1994.

[10] A. M. Cullen and R. M. Mathur, "Real-time, multi-processor, simulation of electromagnetic transients of transmission lines with frequency dependent parameters", *Proc. of The First International Conference on Digital Power System Simulators (ICDS),* Texas, U.S.A., pp. 271-276, April 1995.

[11] A. EI Hakimi, H. Le-Huy, C. Dufour, and I. Kamwa, "Real-time methods for power transmission networks simulation", *Proc. of The First International Conference on Digital Power System Simulators (ICDS),* Texas, U.S.A., pp. 1-4, April 1995.

[12] C. Dufour, H. Le-Huy, J. Soumagne, and A. E. Hakimi, "Real-time simulation of power transmission lines using Marti model with optimal fitting on dual-DSP card", *IEEE Trans. on Power Delivery,* vol. 11, no. 1, pp. 412-419, January 1996.

[13] X. Wang, D. A. Woodford, R. Kuffel, and R. Wierckx, "A real-time transmission line model for a digital TNA", *IEEE Trans. on Power Delivery,* vol. 11, no. 2, pp. 1092-1097, April 1996.

[14] J. R. Marti and L. R. Linares, "Real-time EMTP-based transients simulation", *IEEE Trans. on Power Systems,* vol. 9, no. 3, pp. 1309-1317, August 1994.

[15] Y. Fujimoto, Y. Bin, H. Taoka, H. Tezuka, S. Sumimoto, and Y. Ishikawa, "Real-time power system simulator on a PC-cluster", *Proc. of the International Conference on Power Systems Transients (IPST 1999),* Budapest, Hungary, pp. 671-676, June 1999.

[16] J. A. Hollman and J. R. Marti, "Real-time network simulation with PC-cluster", *IEEE Trans. on Power Systems,* vol. 18, no. 2, pp. 563-569, May 2003.

[17] L. Pak, M. O. Faruque, X. Nie, and V. Dinavahi, "A versatile cluster-based real-time digital simulator for power engineering research", *IEEE Trans. on Power Systems,* vol. 21, no. 2, pp. 455-465, May 2006.

[18] P. G. Mclaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt, "A real time digital simulator for testing relays", *IEEE Trans. on Power Delivery,* vol. 7, no. 1, pp. 207-213, January 1992.

[19] P. Forsyth, T. Maguire, and R. Kuffel, "Real-time digital simulation for control and protection system testing", *The* 35^{th} *Annual IEEE Electronics Specialists Conference,* Aachen, Germany, pp. 1-6, 2004.

[20] Online available at "http://www.rtds.com"

[21] V. Q. Do, J. C. Soumagne, G. Sybille, G. Turmel, P. Giroux, G. Cloutier, and S. Poulin, "HYPERSIM, an integrated real-time simulator for power network and control systems", *Proc. of International Conference on Digital Power System Simulators (ICDS 1999),* Vasteras, Sweden, May 1999.

[22] D. Pare, G. Turmel, J.-C. Soumagne, V. Q. Do, S. Casoria, M. Bissonnette, B. Marcourx, and C. McNabb, "Validation tests of the HYPERSIM digital real time simulator with a large AC-DC network", *Proc. of the International Conference on Power Systems Transients (IPST 2003),* New Orleans, Louisiana, USA, pp. 1-6, September 2003.

[23] C. Larose, S. Guerette, F. Guay, A. Nolet, T. Yamamoto, H. Enomoto, Y. Kono, Y. Hasegawa, and H. Taoka "A fully digital real-time power system simulator based on PC-cluster", *Proc. of Mathematics and Computers in Simulation,* vol. 63, pp. 151-159, November 2003.

[24] J. Bélanger, V. Lapointe, C. Dufour, and L. Schoen, "eMEGAsim: An open high-performance distributed real-time power grid simulator, architecture and specification", *Proc. of the International Conference on Power Systems (ICPS 2007),* Bangalore, India, December 2007.

[25] J. Bélanger, L. A. Snider, J. N. Paquin, C. Pirolli, and W. Li, "A modern and open real-time digital simulator of contemporary power systems", *Proc. of the International Conference on Power Systems Transients (IPST 2009),* Kyoto, Japan, June 2009.

[26] R. Krebs and O. Ruhle, "NETOMAC real-time simulator - a new generation of standard test modules for enhanced relay testing", *Proc. of the IEEE International Conference on Developments in Power System Protection,* Vol. 2, pp. 669-674, April 2004.

[27] B. Kulicke, E. Lerch, O. Ruhle, and W. Winter, "NETOMAC - calculating, analyzing and optimization the dynamic of electrical systems in time and frequency domain", *Proc. of International Conference on Power Systems Transients (IPST 1999),* Budapest, Hungary, pp. 1-6, June 1999.

[28] A. S. Morched and V. Brandwajn, "Transmission network equivalents for electromagnetic transients studies", *IEEE Trans. on Power Apparatus and Systems,* vol. PAS-102, pp. 2984-2994, September 1983.

[29] M. Abdel-Rahman, A. Semlyen, and R. Iravani, "Two-layer network equivalent for electromagnetic transients", *IEEE Trans. on Power Delivery,* vol. 18, no. 4, pp. 1328-1335, October 2003.

[30] X. Nie, Y. Chen, and V. Dinavahi, "Real-time transient simulation based on a robust two-layer network equivalent", *IEEE Trans. on Power Systems,* vol. 22, no. 4, pp. 1771-1781, November 2007.

[31] F. Ricci and H. Le-Huy, "An FPGA-based rapid prototyping platform for variable-speed drives", *Proc. of IEEE 28th Annual Conference of the Industrial Electronics Society (IECON 02),* vol. 2, no., pp. 1156-1161, November 2002.

[32] P. Le-Huy, S. Guerette, L. A. Dessaint, and H. Le-Huy, "Dual-Step Real-Time Simulation of Power Electronic Converters Using an FPGA", *Proc. of IEEE 2006 IEEE International Symposium on Industrial Electronics,* vol. 2, no., pp. 1548-1553, July 2006.

[33] O. A. Mohammed, N. Y. Abed, and S. C. Ganu, "Real-time simulations of electrical machine drives with hardware-in-the-Loop", *Proc. of the IEEE Power Engineering Society General Meeting (2007),* vol., no., pp.1-6, June 2007.

[34] E. Duman, H. Can, and E. Akin, "Real time FPGA implementation of induction machine model - a novel approach", *Proc. of the International Aegean Conference on Electrical Machines and Power Electronics (ACEMP 2007),* vol., no., pp. 603-606, September 2007.

[35] H. Chen. S. Sun, D. C. Aliprantis, and J. Zambreno, "Dynamic simulation of electric machines on FPGA boards", *Proc. of the IEEE International Electric Machines and Drives Conference (IEMDC 2009),* vol., no., pp. 1523-1528, May 2009.

[36] T. Kato, K. Inoue, and T. Hashimoto, "Practical modeling and simulation of a power electronic system", *Proc. of the Power Conversion Conference (PCC 2007),* vol., no., pp. 477-484, Nagoya, April 2007.

[37] G. Parma, and V. Dinavahi, "Real-Time Digital Hardware Simulation of Power Electronics and Drives", *IEEE Trans. on Power Delivery,* vol. 22, no. 2, pp. 1235-1246, April 2007.

[38] J. Poon, P. Haessig, J. G. Hwang, and I. Celanovic, "High-speed hardware-in-the loop platform for rapid prototyping of power electronics systems", *Proc. of the IEEE Conference on Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES 2010),* vol., no., pp. 420-424, September 2010.

[39] A. Myaing and V. Dinavahi, "FPGA-based real-time emulation of power electronic systems with detailed representation of device characteristics", *IEEE Trans. on Industrial Electronics,* vol. 58, no. 1, pp. 358-368, January 2011.

[40] E. Monmasson and M. N. Cirstea, "FPGA design methodology for industrial control systems-a review", *IEEE Trans. on Industrial Electronics,* vol. 54, no. 4, pp. 1824-1842, August 2007.

[41] M. N. Cirstea and A. Dinu, "A VHDL holistic modeling approach and FPGA implementation of a digital sensorless induction motor control scheme", *IEEE Trans. on Industrial Electronics,* vol. 54, no. 4, pp. 1853-1864, August 2007.

[42] Y. F. Chan, M. Moallem, and W. Wang, "Design and implementation of modular FPGA-based PID controllers", *IEEE Trans. on Industrial Electronics,* vol. 54, no. 4, pp. 1898-1906, August 2007.

[43] Y.-S. Kung, C.-C. Huang, and M.-H. Tsai, "FPGA realization of an adaptive fuzzy controller for PMLSM drive", *IEEE Trans. on Industrial Electronics,* vol. 56, no. 8, pp. 2923-2932, August 2009.

[44] L. Idkhajine, E. Monmasson, M. W. Naouar, A. Prata, and K. Bouallaga, "Fully integrated FPGA-based controller for synchronous motor drive", *IEEE Trans. on Industrial Electronics,* vol. 56, no. 10, pp. 4006-4017, October 2009.

[45] S. Karimi, P. Poure, and S. Saadate, "An HIL-based reconfigurable platform for design, implementation, and verification of electrical system digital controllers", *IEEE Trans. on Industrial Electronics,* vol. 57, no. 4, pp. 1226-1236, April 2010.

[46] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan, and M. W. Naouar, "FPGAs in industrial control applications", *IEEE Trans. on Industrial Informatics,* vol. 7, no. 2, pp. 224-243, May 2011.

[47] R. Foley, R. Kavanagh, W. Marnane, and M. Egan, "Multiphase digital pulsewidth modulator", *IEEE Trans. on Power Electronics,* vol. 21, no. 3, pp. 842-846, May 2006.

[48] S. C. Huerta, A. de Castro, O. Garcia, and J. A. Cobos, "FPGA-based digital pulsewidth modulator with time resolution under 2 ns", *IEEE Trans. on Power Electronics,* vol. 23, no. 6, pp. 3135-3141, November 2008.

[49] G. Oriti and A. L. Julian, "Three-phase VSI with FPGA-based multisampled space vector modulation", *IEEE Trans. on Industry Applications,* vol. 47, no. 4, pp. 1813-1820, July 2011.

[50] J. Alvarez, O. Lopez, F. D. Freijedo, and J. Doval-Gandoy, "Digital parameterizable VHDL module for multilevel multiphase space vector PWM", *IEEE Trans. on Industrial Electronics,* vol. 58, no. 9, pp. 3946-3957, September 2011.

[51] S. P. Valsan and K. S. Swarup, "Protective relaying for power transformers using field programmable gate array", *IET Electric Power Applications,* vol. 2, no. 2, pp. 135-143, March 2008.

[52] X. Liu, A. H. Osman, and O. P. Malik, "Real-time implementation of a hybrid protection scheme for bipolar HVDC line using FPGA", *IEEE Trans. on Power Delivery,* vol. 26, no. 1, pp. 101-108, January 2011.

[53] Online documents available at "http://www.altera.com"

[54] Z. Luo and M. Martonosi, "Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques", *IEEE Transactions on Computers,* vol. 49, no. 3, pp. 208-218, March 2000.

[55] A. Semlyen and A. Dabuleanu, "Fast and accurate switching transient calculations on transmission lines with ground return using recursive convolutions", *IEEE Trans. on Power Apparatus and Systems,* Vol. PAS-94, No. 2, pp. 561-571, March/April 1975.

[56] J. R. Marti, "Accurate modeling of frequency-dependent transmission lines in electromagnetic transient simulations", *IEEE Trans. on Power Apparatus and Systems,* vol. PAS-101, no. 1, pp. 147-155, January 1982.

[57] Y. Chen and V. Dinavahi, "FPGA-based real-time EMTP", *IEEE Trans. on Power Delivery,* vol. 24, no. 2, pp. 892-902, April 2009.

[58] H. W. Dommel, "Nonlinear and time-varying elements in digital simulation of electromagnetic transients", *IEEE Trans. on Power Apparatus and Systems,* vol. 90, no. 4, pp. 2561-2567, June 1971.

[59] L. O. Chua, "Efficient computer algorithms for piecewise linear analysis of resistive nonlinear networks", *IEEE Trans. on Circuit Theory,* vol. CT-18, no. 1, pp. 73-85, January 1971.

[60] S. Lachowicz and H. Pfleiderer, "Fast evaluation of the square root and other nonlinear functions in FPGA", *Proc. of 4th IEEE International Symposium on Electronic Design Teat and Applications,* Hongkong, China, pp. 1-4, January 2008.

[61] N. Boullis, O. Mencer, W. Luk, and H. Styles, "Pipelined function evaluation on FPGAs", *Proceedings of 9th IEEE Field-Programmable Custom Computing Machines,* pp. 1-3, 2001.

[62] L. Zhuo and V. K. Prasanna, "Sparse matrix-vector multiplication on FPGAs", *FPGA'05,* Monterey, California, USA, pp. 1-12, February, 2005.

[63] Y. Chen and V. Dinavahi, "An iterative real-time nonlinear electromagnetic transient solver on FPGA", *IEEE Trans. on Industrial Electronics,* vol. 58, no. 6, pp. 2547-2555, June 2011.

[64] H. K. Lauw and W. Scott Meyer, "Universal machine modeling for the representation of rotating electric machinery in an electromagnetic transients program", *IEEE Trans. on Power Apparatus and Systems,* vol. 101, no. 6, pp. 1342-1350, June 1982.

[65] H. K. Lauw, "Interfacing for universal multi-machine system modeling in an electromagnetic transients program", *IEEE Trans. on Power Apparatus and Systems,* vol. 104, no. 9, pp. 2367-2373, September 1985.

[66] L. Marti, "Simulation of transients in underground cables with frequency-dependent modal transformation matrices", *IEEE Trans. on Power Delivery*, vol. 3, no. 3, pp. 1099-1110, July 1988.

[67] T. Noda, N. Nagaoka, and A. Ametani, "Phase domain modeling of frequency-dependent transmission lines by means of an ARMA model", *IEEE Trans. on Power Delivery*, vol. 11, no. 1, pp. 401-411, January 1996.

[68] H. V. Nguyen, H. W. Dommel, and J. R. Marti, "Direct phase-domain modeling of frequency-dependent overhead transmission lines", *IEEE Trans. on Power Delivery*, vol. 12, no. 3, pp. 1335-1342, July 1997.

[69] A. Morched, B. Gustavsen, and M. Tartibi, "A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables", *IEEE Trans. on Power Delivery*, vol. 14, no. 3, pp. 1032-1038, July 1999.

[70] B. Gustavsen and A. Semlyen, "Simulation of transmission line transients using vector fitting and modal decomposition", *IEEE Trans. on Power Delivery*, vol. 13, no. 2, pp. 605-614, April 1998.

[71] Y. Chen and V. Dinavahi, "Digital hardware emulation of universal machine and universal line models for real-time electromagnetic transient simulation", *IEEE Trans. on Industrial Electronics*, pp. 1-8, 2011.

A

System Data of Case Study in Chapter 3

The system data for the real-time simulation case study is given in Table A.1, Table A.2, and Table A.3. Fig. A.1 shows the tower geometry for the lines used in the case study.

Table A.1: Transmission line parameters

Line	Length	Z_c order	A_1 order
L1	120km	12	19
L2	136km	11	19
L3	165km	12	19
L4	150km	11	18
L5	120km	11	20
L6	400km	12	30
L7	220km	12	14
L8	35km	12	24
L9	10km	12	18
L10	35km	12	24
L11	15km	12	12
L12	65km	12	19
L13	133km	12	19
L14	42km	12	21
L15	375km	12	30

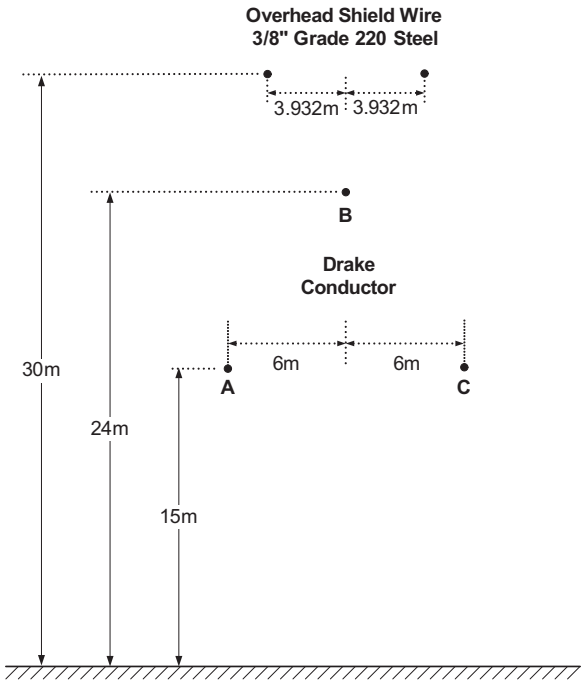


Figure A.1: Tower geometry of transmission lines in the case study.

Table A.2: Load parameters

Load	Z_{ph}
Load1	1200Ω, 500mH
Load2	2150Ω, 380mH
Load3	250Ω, 25mH
Load4	350Ω, 60mH
Load5	250Ω, 25mH
Load6	420Ω, 30mH
Load7	200Ω, 130mH
Load8	650Ω, 250mH
C1	5μF
C2	20μF

Table A.3: Generator and transformer parameters

Generator	Parameters per phase
G1	1.03$\angle$20.2°, Z_{G1}: 1.2Ω, 38.98mH
G2	1.01$\angle$10.5°, Z_{G2}: 1.1Ω, 45.52mH
G3	1.03$\angle$-6.8°, Z_{G3}: 0.9Ω, 38.98mH
G4	1.01$\angle$-17.0°, Z_{G4}: 0.8Ω, 35.23mH
Transformer	Parameters per phase
T1	Z_{T1}: 1.5Ω, 23.4mH
T2	Z_{T2}: 0.8Ω, 29.5mH
T3	Z_{T3}: 1.6Ω, 23.4mH
T4	Z_{T4}: 0.6Ω, 20.8mH

B

System Data of Case Studies in Chapter 4

B.1 Case Study I

Table B.1: Data for Case Study I

Parameters	Values
V_s	20Kv peak
R_s, L_s	5Ω, 24.1mH
Transmission line resistance	mode +: 1.273e-5Ω/m, mode 0: 3.864e-4Ω/m
Transmission line inductance	mode +: 9.337e-4mH/m, mode 0: 4.126e-3mH/m
Transmission line capacitance	mode +: 1.274e-5μF/m, mode 0: 7.751e-6μF/m
Transmission line length	Line1: 100km, Line2: 100km
Series compensator C	100μF
R_{L1}, L_{L1}	8Ω, 4000mH
R_{L2}	270Ω
R_f	0.0001Ω
V_{ref}, p, and q	8192V, 600A, and 6

B.2 Case Study II

Table B.2: Data for Case Study II

Parameters	Values
V_s	125Kv peak
R_s, L_s	1Ω, 15mH
C_W	5nF
C_S	1.25nF
R_{fe}	200MΩ

C

System Data of Case Study in Chapter 5

C.1 Transmission Lines

ULM transmission lines ($Line_1$ and $Line_2$) parameters:

3 conductors, diameter: 3.105cm, resistance: 0.0583Ω/Km, 150Km, and the tower geometry is shown in Fig. C.1.

C.2 Synchronous Machines

UM synchronous machines (UM_1, UM_2, and UM_3) parameters:

1000MVA, 22kV, Y connected, field current: 2494A, 2 poles, 60Hz, single-mass, J=5.5628e4 [$kg - m^2$], D=0.678e4 [$Nm - s/rad$].

The winding resistances (in Ω) and leakage inductances (in H) are listed in Table C.1.

C.3 Loads and Transformers

Loads and transformers parameters:

$Load_1, Load_2$, and $Load_3$: 5Ω, 4H

X_{T1}, X_{T2}, and X_{T3}: 15mH

C: 20μF

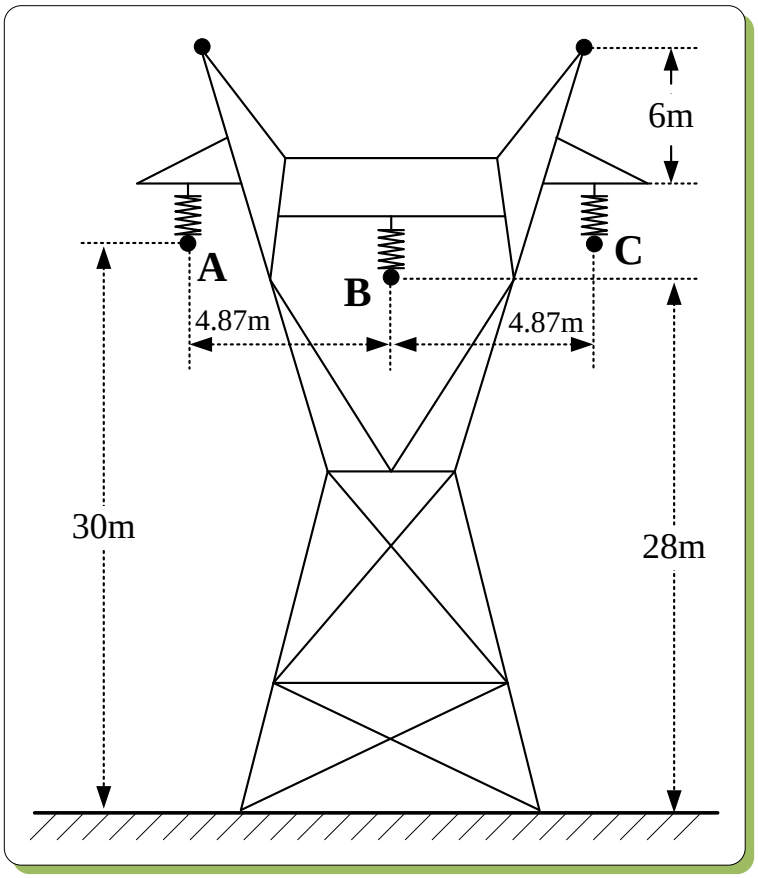


Figure C.1: Tower geometry of transmission lines in the case study.

Table C.1: UM machine resistances and inductances

R_d	9.6800e-4	R_q	9.6800e-4	R_0	9.6800e-4
R_f	1.1109	R_{D1}	3.4985	R_{Q1}	0.7627
R_{Q2}	1.2270				
L_d	0.0018	L_q	0.0017	L_0	2.4136e-4
L_f	0.6345	L_{D1}	0.5483	L_{Q1}	1.1811
L_{Q2}	0.5882	M_{df}	0.0234	M_{dD1}	0.0234
M_{qQ1}	0.0226	M_{qQ2}	0.0226	M_{fD1}	0.5480
M_{Q1Q2}	0.4184				